A High-Performance Hardware Architecture of An Image Matching System Based on the Optimised SIFT Algorithm

WENJUAN DENG, BEng.

Department of Electrical and Electronic Engineering

Thesis submitted to the University of Nottingham

for the degree of Doctor of Philosophy

September, 2013

Abstract

The Scale Invariant Feature Transform (SIFT) is one of the most popular matching algorithms in the field of computer vision. It has advantages over many other algorithms because features detected are fully invariant to image scaling and rotation, and are also shown to be robust to changes in 3D viewpoint, addition of noise, changes in illumination and a sustainable range of affine distortion. However, the computational complexity is high, which prevents it from achieving real-time performance. The aim of this project, therefore, is to develop a high-performance image matching system based on the optimised SIFT algorithm to perform real-time feature detection, description and matching. This thesis presents the stages of the development of the system.

To reduce the computational complexity, an alternative to the grid layout of standard SIFT is proposed, which is termed as SRI-DAISY (Scale and Rotation Invariant DAISY). The SRI-DAISY achieves comparable performance with the standard SIFT descriptor, but is more efficient to be implemented using hardware, in terms of both computational complexity and memory usage. The design takes only 7.57 μs to generate a descriptor with a system frequency of 100 MHz, which is equivalent to approximately 132,100 descriptors per second and is of the highest throughput when compared with existing designs. Besides, a novel keypoint matching strategy is also presented in this thesis, which achieves higher precision than the widely applied distance ratio based matching and is computationally more efficient. All phases of the SIFT algorithm have been investigated, including feature detection, descriptor generation and descriptor matching. The characterisation of each individual part of the design is carried out and compared with the software simulation results.

A fully stand-alone image matching system has been developed that consists of a CMOS camera front-end for image capture, a SIFT processing core embedded in a Field Programmable Logic Array (FPGA) device, and a USB back-end for data transfer. Experiments are conducted by using real-world images to verify the system performance. The system has been tested by integrating into two practical

applications. The resulting image matching system eliminates the bottlenecks that limit the overall throughput of the system, and hence allowing the system to process images in real-time without interruption. The design can be modified to adapt to the applications processing images with higher resolution and is still able to achieve real-time.

Acknowledgement

I would like to express my sincere appreciation to my supervisor Dr. Yiqun Zhu for all his guidance and patience throughout this project. I learn from him not only the knowledge but also the philosophy of scientific research.

I would also like to thank Hao Feng for his valuable discussion on the theory and Lifan Yao for his discussion on embedded system at the beginning of this project. Thanks are also due to Prof. Stephen P. Morgan for his valuable comments on the academic writing of annual reports. I am also grateful to my colleagues and technicians for their help during my doctoral study.

I would like to acknowledge the University of Nottingham and the China Scholarship Council (CSC) for the financial support.

I would especially like to thank my family for the endless support and encouragement. I am also grateful to my friends for rendering my PhD time colourful. Finally, thanks go to Hao Lu for his patience and love.

List of Publications

Wenjuan Deng, Yiqun Zhu, Hao Feng, and Zhiguo Jiang. An efficient hardware architecture of the optimised SIFT descriptor generation. In: 22nd International Conference on Field Programmable Logic and Application (FPL), 2012, 345-352.

Wenjuan Deng, Yiqun Zhu. A memory-efficient hardware architecture for real-time feature detection of the SIFT algorithm. In: 21st ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), 2013 (accepted).

List of Abbreviations

1D	One-Dimensional
2D	Two-Dimensional
BRAM	Block RAM
CMOS	Complementary Metal Oxide Semiconductor
DPRAM	Dual Port RAM
EDK	Embedded Development Kit
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
GMO	Gradient Magnitude and Orientation
GPIO	General Purpose Input / Output
HDL	Hardware Description Language
LSB	Least Significant Bit
LUT	Look Up Table
MSB	Most Significant Bit
PC	Personal Computer
PCB	Printed Circuit Board
PLB	Processor Local Bus
PU	Processing Unit
QVGA	Quarter-VGA
RAM	Random Access Memory
SDK	Software Development Kit

SIFT	Scale Invariant Feature Transform
SRT	Shift RegisTer
SVGA	Super-VGA
VGA	Video Graphic Array
HDL	Hardware Description Language
XGA	eXtended Graphics Array
XPS	Xilinx Platform Studio

Table of Contents

Chapter 1 Introduction

1.1 Introduction

Computer vision, which focuses on providing computers with the ability to mimic human perception, constitutes an important step in designing systems which can perform intelligent tasks.

Object or scene recognition is one of the fundamental tasks in the field of computer vision. One of the frequently used contexts is to identify the presence of specific objects or a class of objects along with their locations in the scene. Recognition is also used in identification of a wide variety of patterns, such as fingerprints and faces just to name a few. Besides, it is an important part of applications such as image retrieval, where the objective is to find an image similar to a given query image.

A common step in most recognition algorithms is to represent the image content in terms of features. A local feature, which is also known as an interest point, is an image pattern that is associated with a change of an image property neighbourhood, such as intensity, colour and texture. Local features can be points, edges and small image patches. In general, a good feature should have the following properties:

- *Repeatability*: The repeatability represents the percentage of points that are simultaneously presented in the commonly visible part of two images that are taken under different viewing conditions. A high repeatability is expected.
- *Distinctiveness*: The feature should show a lot of variations of the local intensity pattern underlying the feature, so that the features can survive large image transformations and hence can be correctly matched.
- *Locality*: The regions identified by features of higher locality are less likely to be occluded or suffer from geometric and photometric transformations between two images taken under different viewing conditions. However, the disadvantage is that the detected regions contain less information and are less distinguished to survive large transformation. Therefore, the keypoints with high locality are less likely to be repeatedly detected and corrected matched, especially in existing of large transformation between images.
- *Quantity*: The number of features should be sufficiently large to meet the requirement of different applications. Ideally, the features should densely cover the entire image. This property is especially useful in applications, such

as object or scene recognition, where it is vital to have features densely covering the entire object of interest. If too few features are detected, the image content is not reliably represented.

- *Accuracy*: The features should be accurately localised in 2D image plane and also in scale space. This property is important for applications, such as wide-base line matching and camera calibration, where accurate locations are needed.

Of all the above mentioned expected properties from local features, repeatability is the most important one and has been widely used in the performance evaluation of detectors [1].

Image matching is an important aspect of computer vision and has been widely used in solving problems related to object or scene recognition [2] [3], robot localisation and mapping [4] [5], object tracking [6] [7], 3D vision [8] [9] and etc. It obtains the similarity of image pairs by identifying their relationship. In general, the image matching usually involves three important stages.

Detection: The first one is the extraction of salient keypoints from images, where each keypoint is typically associated with information, such as the location in 2D image plane and scale space.

Description: The second stage is to associate each keypoint with a distinctive descriptor based on the local region around the keypoint.

Matching: The final stage is the matching of keypoints between images based on the descriptors.

1.2 Motivation

In the past few decades, a considerable amount of research has been made to explore effective algorithms to determine correspondence between images. SIFT (Scale Invariant Feature Transform) [10] has advantages over other algorithms because features detected are fully invariant to image scaling and rotation, and are partially invariant to changes in 3D viewpoint, addition of noise, and changes in illumination. However, the high computational complexity makes it not eligible to real-time

applications. In recent years, an impressive body of work has been done to improve both the efficiency and performance of the standard SIFT algorithm. Apart from developing variations to the standard SIFT algorithm, efforts have been made to explore pipelined hardware architecture while seeking for help from new hardware technologies. Related researches will be reviewed in Chapter 2, in terms of SIFT variations and efficient hardware implementations.

1.3 Objectives and Contributions

The research presented in this thesis aims at tackling the major drawback of the existing systems, which is the relatively low overall processing throughput with feature description incorporated, and hence providing a high frame rate and high accuracy image matching system.

This research mainly consists of two parts:

- The theoretical part, such as the improved spatial arrangement of descriptor, and the parameters that can be tuned to improve hardware efficiency while keeping relatively high performance.
- The hardware part, such as the hardware architecture of the SIFT processing core, and the complete image matching system.

The main objectives of this project are:

- Appropriate system configuration and algorithm modification for an efficient hardware design.
- High frame rate image processing system. The ultimate target for the frame rate is 60 fps for VGA images.
- High accuracy processing core so that the matching performance is comparable with the high-precision software model.
- Low resource usage so that the processing core can be integrated into a single chip, which means the whole system on a chip (SoC).

The main contributions of this project are:

- The grid layout of the standard SIFT descriptor is optimised by using the log-polar spatial arrangement, which is more efficient to compute without significant performance degradation.
- A novel feature matching strategy is proposed, which provides higher matching accuracy when compared with existing widely applied matching methods.
- A *rotating buffer* memory solution is proposed, with which the memory requirement remains constant with the increase of the parallelism level of the processing core and it contributes to the memory reduction of the design.
- A fully stand-alone image matching system is developed, which achieves real-time performance for VGA images and is the first complete hardware design for the SIFT algorithm.

1.4 Thesis Outline

This thesis presents the research carried out to achieve a real-time image matching system of high frame rate and low hardware resource usage based on the optimised SIFT Algorithm. The remainder of the thesis is organised as follows.

In Chapter 2, a review to the related researches is presented, in terms of both the image matching algorithms and the existing approaches that improve both the efficiency and performance of the standard SIFT algorithm. The drawbacks of the existing hardware systems developed for the SIFT algorithm are presented, which leads to the necessity of this research.

Chapter 3 introduces the optimisations toward the standard 128-dimentional descriptor. Evaluation is performed to compare the performance of the standard SIFT and the spatial arrangement of the descriptor, named SRI-DAISY (Scale and Rotation Invariant DAISY). A novel image matching strategy is proposed in the same chapter, which achieves higher precision than distance ratio based matching from SIFT and is more efficient to implement on hardware devices.

Chapter 4 presents a detailed analysis to the parameters that affect the performance and hardware efficiency of the SIFT processing core. Detailed evaluation is

performed to achieve an appropriate parameter setting for an efficient hardware design.

In Chapter 5, FPGA based hardware architecture of the SIFT processing core is presented, which covers all phases of the optimised SIFT algorithm. Memory requirement is analysed and efficient memory solutions are provided.

Chapter 6 presents the developed embedded system for the optimised SIFT algorithm. Tests and experiments are conducted for the performance evaluation of the system, in terms of robustness to geometric and photometric transformations. Besides, the matching performance is tested in two applications: object recognition and video stabilisation.

Finally, Chapter 7 concludes the thesis and presents discussion and suggestions for further work.

Chapter 2 Image Matching

2.1 Introduction

This chapter mainly consists of three parts. Firstly, relevant research on the intensity based feature detection methods that led to the current state-of-the-art SIFT algorithm are reviewed. Secondly, variations to the SIFT are introduced, which are developed to improve either the efficiency or the performance of the standard SIFT algorithm. Finally, a review of related work to speed up the SIFT implementation is also presented with advantages and disadvantages that led to the necessity of the research reported in this thesis.

2.2 Related Image Matching Algorithms

In this section, relevant researches on the feature detectors are reviewed, and an emphasis is placed on the approaches proposed for extracting scale invariant features that are closely related to the SIFT algorithm.

In this section, two major types of local features are reviewed: corner detector and blob-like structure detector. The corner detector detects corners and highly textured points, whereas the blob-like structure detector detects mainly blobs. A corner can be identified by a single point while a blob relies on the boundary of its neighbourhood. Corners are typically better localised in the image plane, and hence are suitable for applications where localisation accuracy is of great concern, such as camera calibration and estimation of epipolar geometry for wide-baseline matching. The blob-like structures are less accurately localised in the 2D image than corners, because the second derivatives give small response in the point where the signal change is most significant. Therefore, blob-like structure detectors are less suited for applications where precise correspondences are needed. However, since blob-like structure detector gives a good estimation of the size thus the scale of the blob, it is better suited to applications where a precision localisation is not necessary, such as object or scene recognition. In practice, the blob detector is complementary to corner detector, and hence are often used together [11] [12] [13].

2.2.1 Corner Detector

In the traditional sense, the corner refers to a point in the 2D image that has large curvature in both directions. Freeman [14] defined corners as discontinuity of an average curve slope and the mean curvature to either side of it can be considered to be uniform and free of discontinuities. It was then noticed that the so-called corners can also be detected from image locations that have large gradients in all directions, such as a small dark spot on a bright background. Nowadays, the term "corner" is used for both senses.

The development of image matching by using a set of local interest points can be traced back to the work of Moravec [15] on stereo matching using a corner detector, which functions by considering a local window in the image. A corner is detected if the average changes of image intensity resulting from a small amount of window shift are large in all directions. As shown in Figure 2-1, the red square represents the image window $w(x, y)$. The leftmost image shows that the image intensity within the window is approximated constant and window shifts in all directions will result in a small change. The middle image shows an edge, where the window shifts along the edge will result in a small amount of change, while the shifts perpendicular to the edge will result in a large change. The rightmost image shows an actual corner, and the window shifts will result in large changes in all possible directions.

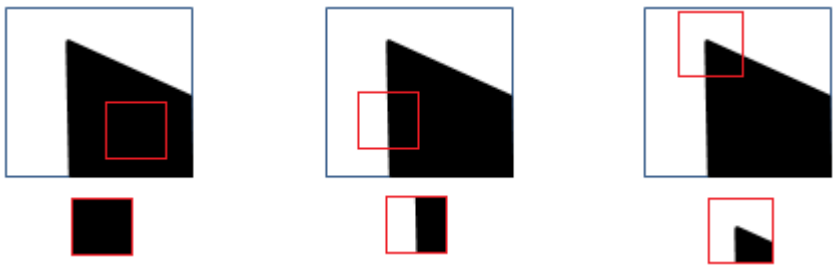


Figure 2-1: Moravec's corner detector.

The similarity between image windows before and after the movement in a certain direction is determined by calculating the sum of squared difference (SSD), as defined in Equation (2.1).

$$E(\mathrm{u},\mathrm{v}) = \sum_{\mathrm{x},\mathrm{y}} w(\mathrm{x},\mathrm{y})|I(\mathrm{x}+\mathrm{u},\mathrm{y}+\mathrm{v}) - I(\mathrm{x},\mathrm{y})|^2 \tag{2.1}$$

where $w(\mathrm{x},\mathrm{y})$ specifies the image window and it is unity within a specified rectangular region and zeros elsewhere. $I(\mathrm{x},\mathrm{y})$ and $I(\mathrm{x}+\mathrm{u},\mathrm{y}+\mathrm{v})$ are the original and shifted pixel intensity, respectively.

Smaller SSD indicates higher similarity and the Moravec's corner detector is actually searching for the minimum $E(\mathrm{u},\mathrm{v})$ in all directions that is above a certain threshold.

The three major drawbacks of the Moravec's corner detector are listed below, which are later improved by the Harris corner detector [16].

1. Shifts in only eight discrete directions are considered, and hence the response is anisotropic.
2. The response is sensitive to noise due to use of binary and rectangular image window.
3. Because it takes into account only the minimum of $E(\mathrm{u},\mathrm{v})$, the detector responds too readily to edges.

One of the intensively used pixel based matching algorithms is developed by Harris and Stephen [16], which is improved upon Moravec's work and is known today as the Harris detector. It concerns not only corners but also any image location that has large gradients in all directions at a predetermined scale. The Harris corner detector is based on the second moment matrix, which is also known as auto-correlation matrix that summarises the gradient distribution in a specified neighbourhood of a point:

$$M = g(\sigma_I) * \begin{bmatrix} I_x^2(\mathbf{x},\sigma_D) & I_x I_y(\mathbf{x},\sigma_D) \\ I_x I_y(\mathbf{x},\sigma_D) & I_y^2(\mathbf{x},\sigma_D) \end{bmatrix} \tag{2.2}$$

with

$$g(\sigma_I) = \frac{1}{2\pi\sigma_I^2} e^{-\frac{x^2+y^2}{2\sigma_I^2}}$$

$$I_x(\mathbf{x}, \sigma_D) = \frac{\partial}{\partial x} g(\sigma_D) * I(\mathbf{x})$$

$$I_y(\mathbf{x}, \sigma_D) = \frac{\partial}{\partial y} g(\sigma_D) * I(\mathbf{x})$$

where σ_D is the differential scale with which the first-order local image derivatives (I_x, I_y) are computed. σ_I is the integration scale of the Gaussian kernel that is applied to the neighbourhood of the pixel to smooth the local image derivatives.

A corner typically has large principal curvature in all directions and can be obtained by analysing the principal curvature in 2D images. Because the eigenvalues of the second moment matrix *M* are proportional to the amount of the principal curvatures of $I(\mathbf{x}, \sigma_D)$, a pixel is labelled a corner if the eigenvalues of the corresponding matrix are both large.

By considering the differential of the corner score with respect to the directions directly instead of using shifted patches, the Harris corner detector removes the anisotropic response limitation of the Moravec's method. The noisy response of Moravec's detector is addressed by using a Gaussian window instead of the square and binary one, which uses a circular window with more weights put on the pixels closer to the centre instead of simple sum in Moravec's method. Finally, the sensitivity to edges is eliminated by analysing the principal curvatures of the local 2D images. An example of detection comparison is shown in Figure 2-2. The left image shows the corners detected with Moravec's method, and the right image shows the corners detected using Harris detector.

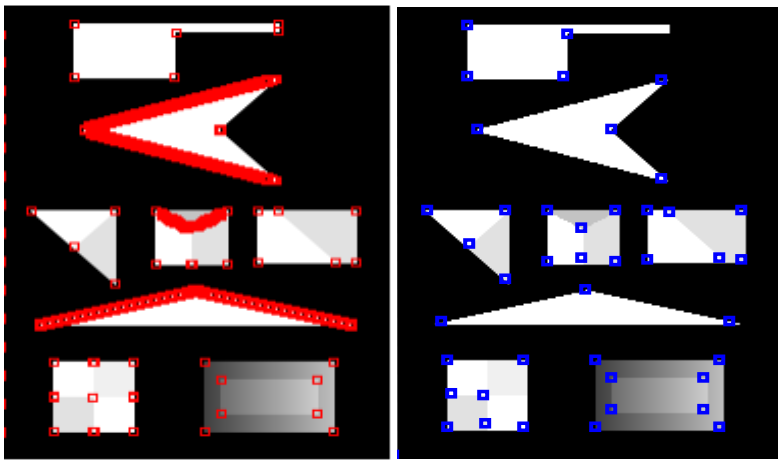


Figure 2-2: An example of detection comparison of Moravec's corner detector and Harris detector.

Harris corner detector is considered as one of the most reliable interest point detectors and is stable in arbitrary lighting conditions. However, it is very susceptible to changes in image scale, therefore fails to provide satisfying matching performance when dealing with images of scales changes, which always occurs in images. To tackle this problem, researches have been done to extract scale invariant features, including improving the detectors to be scale adapted and exploring features that are detected in the scale invariant manner.

A variation of Harris detector is proposed in [17], which is referred to as Harris-Laplace. The Harris-Laplace detector is a combination of Harris detector and Laplace operator proposed by Lindeberg [18]. It starts with the multi-scale point selection using scale adapted Harris corner detector, followed by iterative scale selection using Laplace operator, which works together to detect scale invariant features. The idea of using the Laplace operator is to select the characteristic scale at which the similarity between the detector operator and the local image structure achieves maximum, which can be explained as finding the circular shape of the Laplacian kernel that is adapted to a local image structure. The characteristic scale is an estimation of the characteristic length of the corresponding image structure, and is related to the structure and not to the resolution at which the structure is represented [17]. As shown in Figure 2-3, the top row shows the images of different scales, where the yellow circles represent the corresponding circle of Laplacian kernel. The bottom

row shows the Laplacian responses over scale, and the characteristic scale are 10.1 and 3.89 for the left and right images, respectively [1]. The Laplacian response achieves a maximum when the size of the Laplace operator matches that of the blob-like structure.

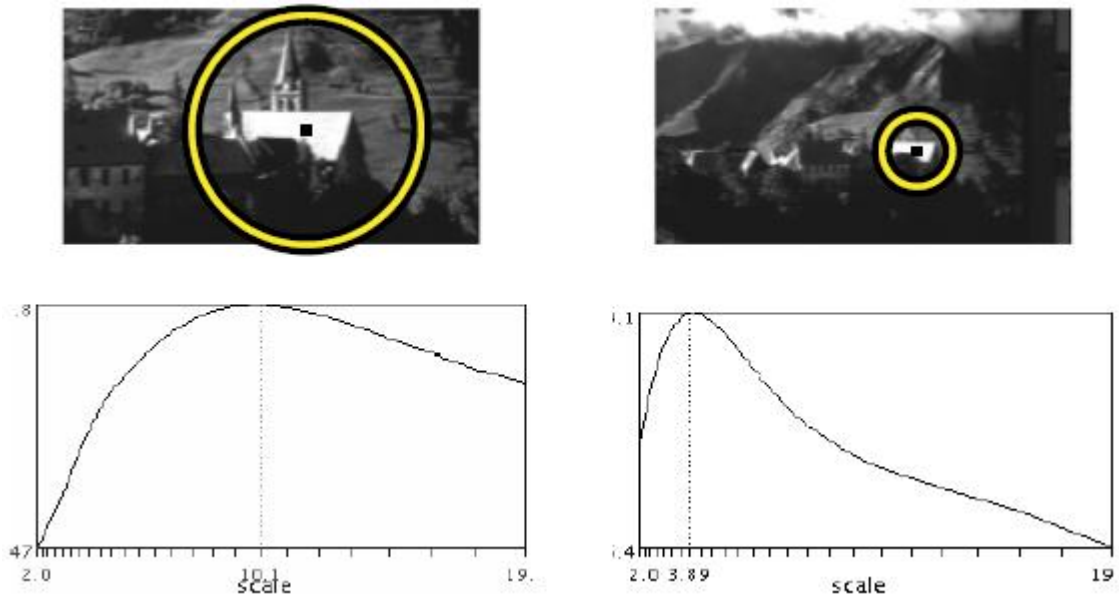


Figure 2-3: An example of characteristic scale selection using Laplace operator [17].

Beaudet [19] proposed a rotation invariant Hessian-based detector termed DET, which is derived from the second-order Taylor expansion of the intensity surface, and especially the Hessian matrix that describes the local curvature. Beaudet defined an operator called DET:

$$\mathrm{DET} = I_{xx}I_{yy} - I_{xy}^2 \tag{2.3}$$

Operator DET is related to the local curvature, and the features correspond to points where DET achieves local extrema.

2.2.2 Blob Detector

Another most intuitive local feature is the blob, which is a region in an image that is either brighter or darker than the surrounding. In this section, three most widely applied blob detectors are reviewed: Laplacian-of-Gaussian (LoG), Determinant-of-Hessian (DoH), and Difference-of-Gaussian (DoG).

a. Laplacian-of-Gaussian

One of the first and most common blob detectors is proposed by Lindeberg [20], which is based on the Laplacian-of-Gaussian (LoG). It searches for extrema from the scale space [21] using the scale normalised LoG operator in Equation (2.4).

$$|\mathrm{LoG}(\mathbf{x}, \sigma_n)| = \sigma_n^2 \left| L_{xx}(\mathbf{x}, \sigma_n) + L_{yy}(\mathbf{x}, \sigma_n) \right| \tag{2.4}$$

where L_{xx} and L_{yy} are the second order derivatives computed using Gaussian kernel of standard deviation σ_n.

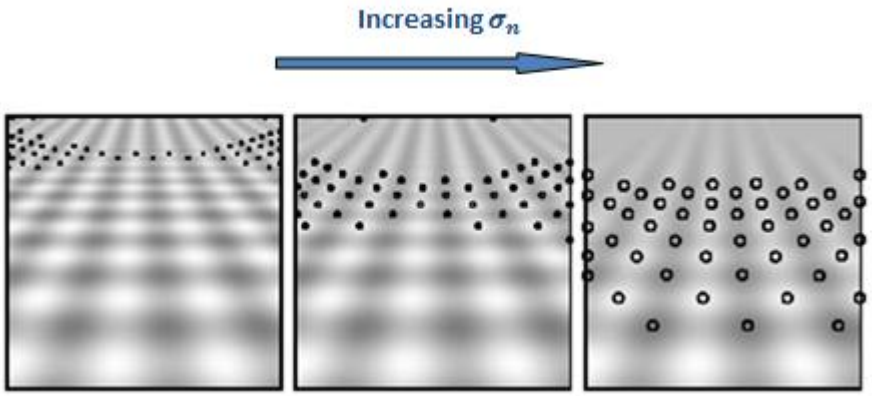
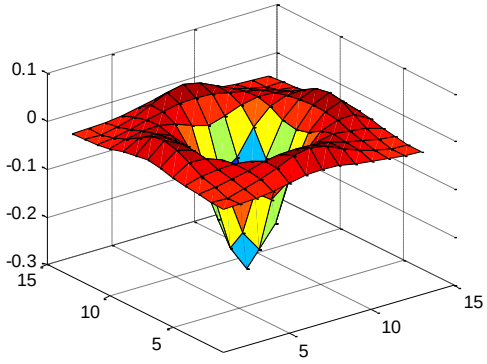
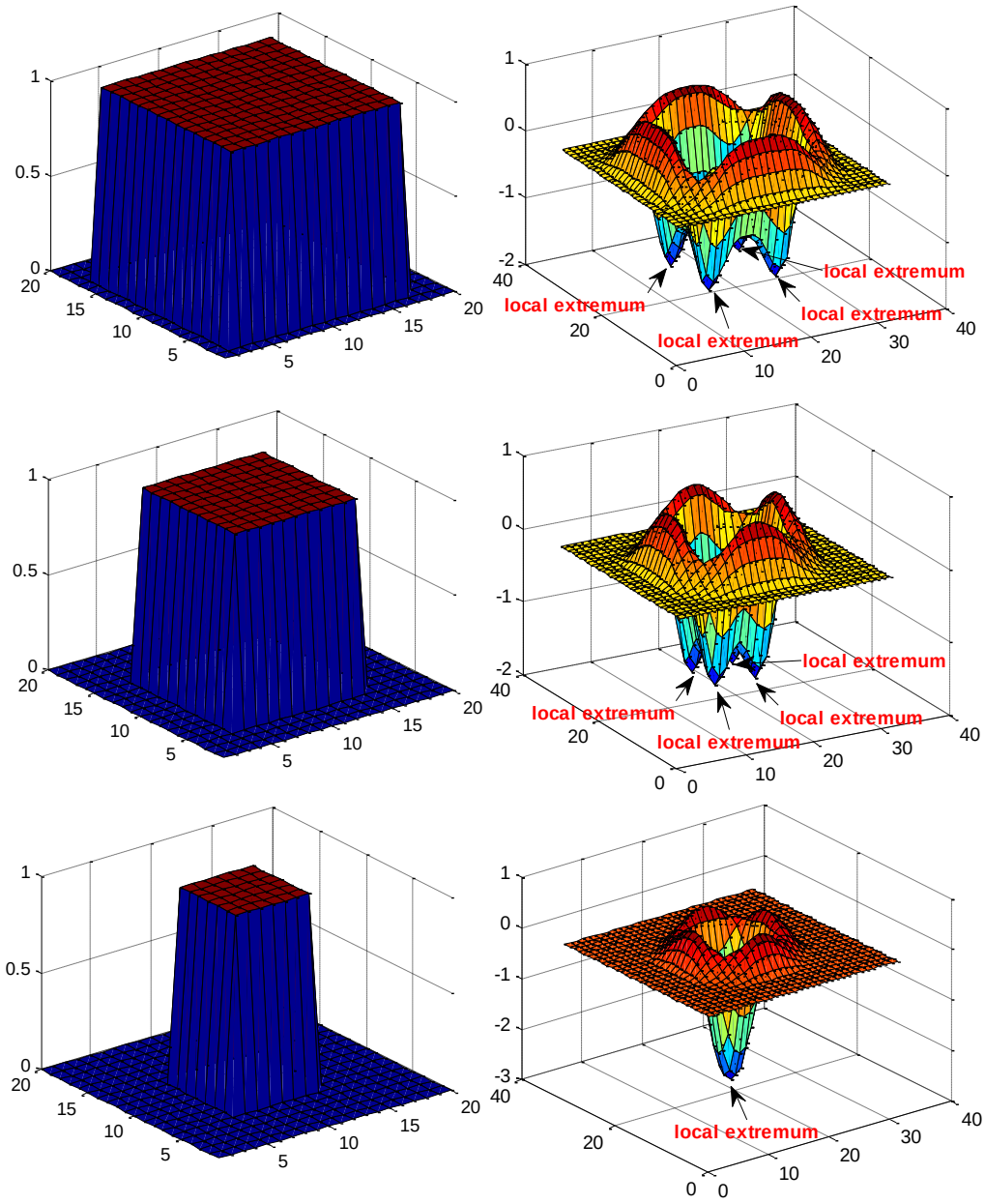


Figure 2-4: Spatial responses to the Laplacian operator computed at different scale levels [20].

Figure 2-4 shows how the spatial responses vary with the Laplacian operator computed at different scale levels. The scale space is generated by successive smoothing of the high resolution image with Gaussian based kernels of different sizes. Koenderink [22] and Lindeberg [23] have shown that Gaussian function is the only possible scale-space kernel.



(a) LoG operator with $\sigma_n = 2.0$.



(b) LoG responses to different signals. Left: signals, Right: LoG responses.

Figure 2-5: LoG operator applied to several different signals.

The LoG detector is able to deal with significant scale changes, but the main drawback is that local maxima are detected from both blob-like structures and the neighbourhood of contours and edges. As shown in Figure 2-5, the Laplacian operator in Figure 2-5(a) responses to edges in the first two examples in Figure 2-5(b), and responses to the blob like structure in the last example. Therefore, to detect a blob, the response of the Laplacian operator should achieve the extrema at the centre of the blob, where a maximum response and minimum response corresponds to a dark blob on light background and light blob on dark background, respectively.

b. Determinant-of-Hessian

A Hessian-based blob-like structure detector is proposed by Mikolajczyk [1], which employs both the trace and determinant of the Hessian matrix (DoH) for feature detection.

$$H = \begin{bmatrix} I_{xx}(\mathbf{x}, \sigma_D) & I_{xy}(\mathbf{x}, \sigma_D) \\ I_{xy}(\mathbf{x}, \sigma_D) & I_{yy}(\mathbf{x}, \sigma_D) \end{bmatrix} \tag{2.5}$$

with

$$I_{xx}(\mathbf{x}, \sigma_D) = \frac{\partial^2}{\partial x^2} g(\sigma_D) * I(\mathbf{x})$$

where $I_{xx}(\mathbf{x}, \sigma_D)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2} g(\sigma_D)$ with the image I at point $\mathbf{x}$, and similarly for $I_{yy}(\mathbf{x}, \sigma_D)$ and $I_{xy}(\mathbf{x}, \sigma_D)$. σ_D is the scale at which the second-order local image derivatives are computed.

The trace of the Hessian matrix is often referred to as Laplacian, which has a strong response to both blob like structures and edges, as has been illustrated in Figure 2-5. A local maximum of DoH indicates the presence of a feature point with large local curvatures. A feature is selected when the trace and DoH simultaneously achieves local extrema, so as to overcome the drawback of the Laplacian which has strong response to edges. As shown in Figure 2-6, the left image shows the detection results

using trace (Laplacian). The right image shows the detection result using both trace and DoH ($\sigma_D = 4$).

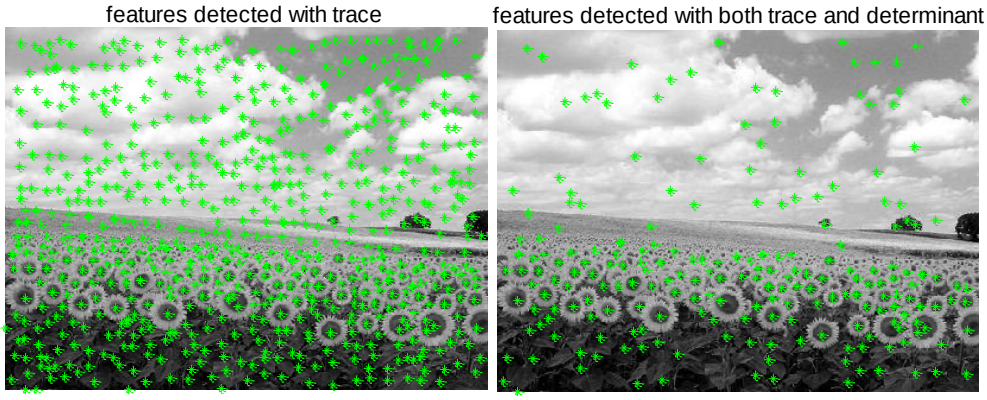
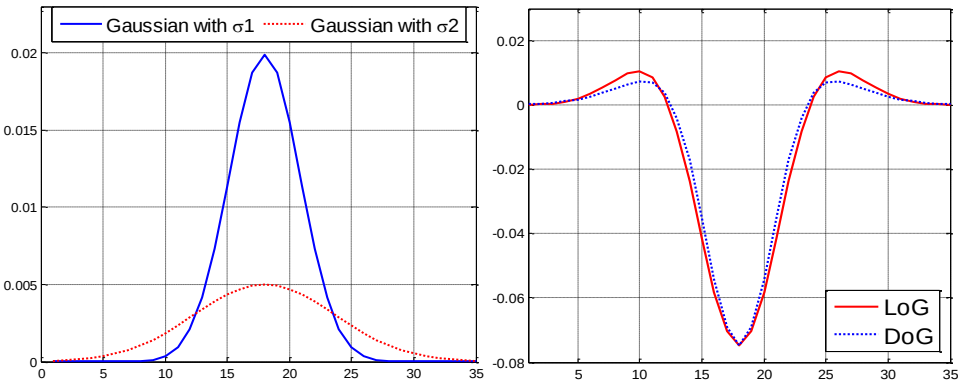


Figure 2-6: Detection results.

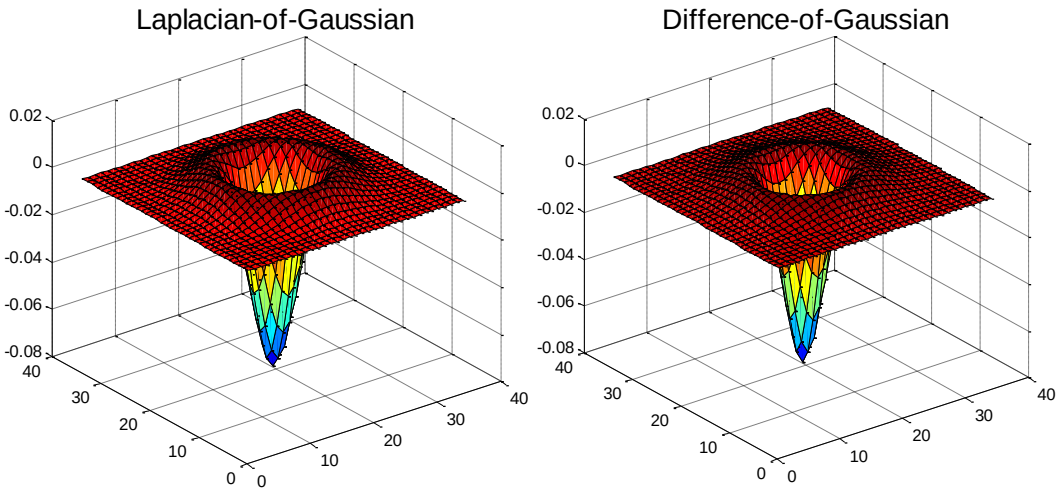
To make Hessian detector to be invariant to scale changes, Hessian-Laplace [17] is developed which are similar to Harris-Laplace, but the features are detected using DoH. According to the comparisons in [11] [18], the Hessian-based detector is more stable and repeatable than Harris-based detectors.

c. Difference-of-Gaussian

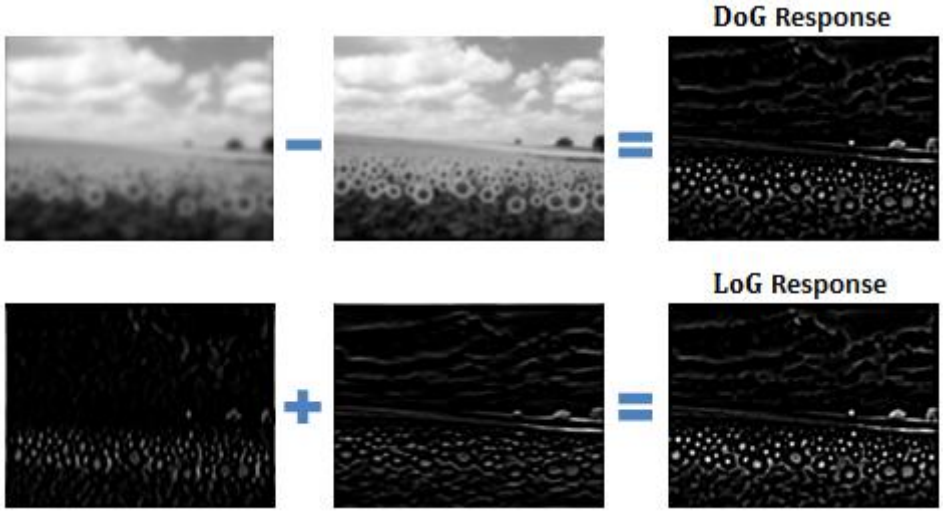
The Difference-of-Gaussian (DoG) has been widely used for feature detection [3] [10] [24] [25] [26], which is a close approximation to the Laplacian-of-Gaussian (LoG) and detects blob-like structures. The DoG represented by the dashed line in the right image of Figure 2-7(b) is generated by applying subtractions to the two Gaussian functions with different standard deviations (σ_1, σ_2) shown in the left image of Figure 2-7(a). The 2D example of DoG and LoG is given in Figure 2-7(b). The DoG is computationally more efficient than LoG, because the second-order derivatives (L_{xx}, L_{yy}) with respect to scale of LoG are approximated by the difference of Gaussian blurred images, as shown in Figure 2-7(c).



(a) 1D example for DoG and LoG.



(b) 2D example for LoG and DoG.



(c) Comparison between DoG response and LoG response

Figure 2-7: Comparison between LoG and DoG.

Lowe [10] extended the DoG operator to deal with scale changes for the SIFT algorithm, which is especially designed for image scaling. In the rest of this section, detailed description of the standard SIFT algorithm is given. The SIFT algorithm mainly consists of two parts: feature detection and descriptor generation. Feature detection module extracts the image features that are further transformed to descriptor vectors in descriptor generation module.

Feature Detection

The feature detection module mainly consists of three stages: 1) Gaussian scale space and DoG space construction. 2) Keypoint detection with stability checking. 3) Gradient Magnitude and Orientation (GMO) calculation.

To achieve invariance to scale change of the image, stable features are searched across all possible scales by using a continuous function of scale known as scale space. The Gaussian scale space is built up by repeatedly convolving the input image I with Gaussian kernel G of different sizes, thereby leading to the scale space composed of a series of smoothed images L of the same resolution at discrete values of σ, as shown in Equation (2.6).

$$\mathrm{L}(x, y, \sigma) = \mathrm{G}(x, y, \sigma) * \mathrm{I}(x, y) \tag{2.6}$$

where * is the convolution operator, σ decides the size of Gaussian kernel given in Equation (2.7).

$$\mathrm{G}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} \tag{2.7}$$

The scale of a scale space image is equal to the standard deviation of Gaussian kernel used to generate that image. Figure 2-8 illustrates the Gaussian scale space and DoG space construction by showing an example of six scales per octave. Scale space images with the same resolution compose an octave. The input image to a new octave is generated by sub-sampling image from the previous octave spatially by a factor of two. The DoG given in Equation (2.8) is generated by applying subtraction operation to adjacent scale space images, which are separated by a constant multiplicative factor *k*.

$$\mathrm{D}(x, y, \sigma) = \mathrm{L}(x, y, k\sigma) - \mathrm{L}(x, y, \sigma) \tag{2.8}$$

A pixel is defined as a keypoint when it is larger or smaller than its 26 neighbours in the DoG space, with eight pixels in the same scale and nine in the scales above and below, respectively. As shown in Figure 2-8, each square represents a pixel in the DoG space, and the pixels in shadow correspond to the pixels to be compared with their neighbouring pixels.

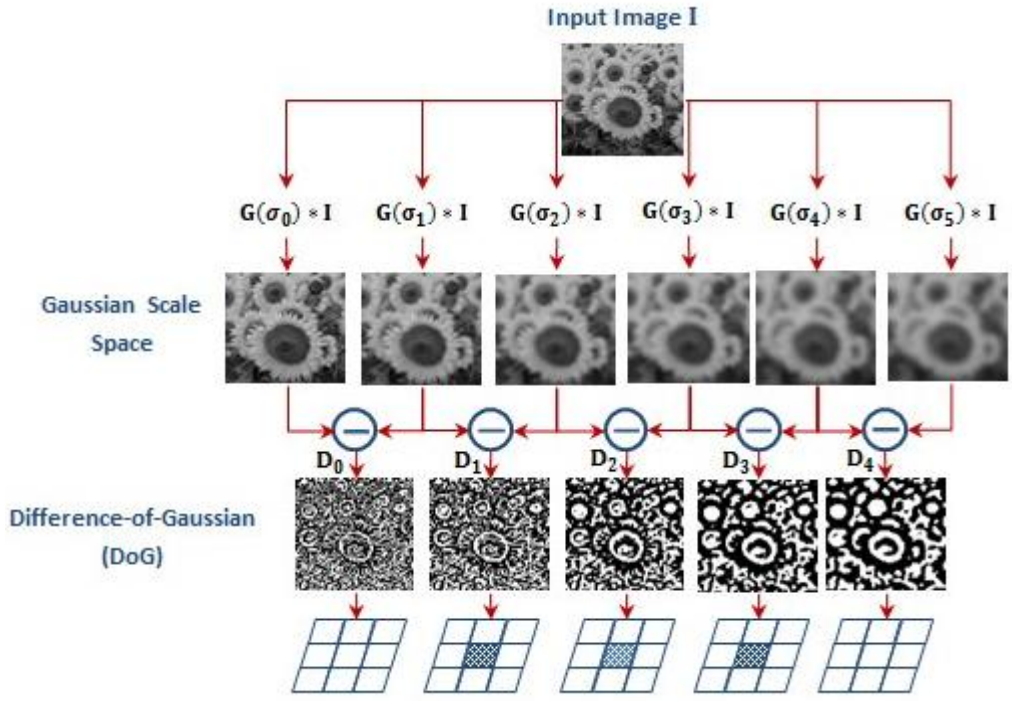


Figure 2-8: Block diagram representing the Gaussian scale space and DoG space construction by using a set of six Gaussian smoothed images.

Once a keypoint has been detected, it will be passed onto the stability checking process to eliminate those that are likely to be unstable, either because they are near an edge rather than a blob-like structure, or because they are found to be with low contrast. Firstly, the location of the keypoint is improved to sub-pixel accuracy by using a second-order Taylor expansion at its original location $u(x, y, \sigma)$. The correction offset w from u with respect to coordinates (x, y) and scale $\sigma$is defined in Equation (2.9).

$$w = -\left(\frac{\partial^2 D}{\partial u^2}\right)^{-1} \frac{\partial D}{\partial u} \tag{2.9}$$

If $w$is larger than 0.5 in any one of the three dimensions (x, y, σ), the keypoint actually lies closer to another pixel, as shown in Figure 2-9. Then the correction offset will be added to u to produce the new location. This process repeats until w is smaller than 0.5 in all dimensions.

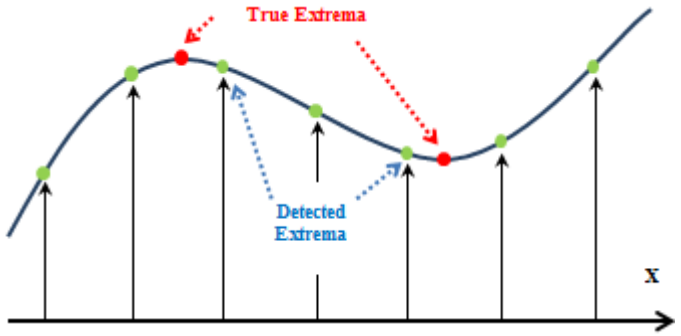


Figure 2-9: Keypoint localisation under sub-pixel accuracy.

Secondly, keypoint with contract c defined in Equation (2.10) lower than the pre-defined threshold is rejected to improve the stability of the system.

$$c = D + \frac{1}{2}\frac{\partial D^T}{\partial u} w \tag{2.10}$$

The final step of stability checking process is the principal curvature analysis. Because the DoG responses to both blob-like structures and edges, the principal curvature analysis step is necessary so as to remove local extrema that are located along edges. This step is achieved by evaluating the eigenvalues of the corresponding Hessian matrix. A local extrema is accepted if:

$$\frac{\mathrm{Tr}(\mathbf{H})^2}{\mathrm{Det}(\mathbf{H})} = \frac{(D_{xx} + D_{yy})^2}{D_{xx}D_{yy} - D_{xy}D_{xy}} < Threshold \tag{2.11}$$

where $\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$.

Local extrema that have passed through all these steps can be identified as keypoints with high confidence.

Descriptor Generation

The next step is to associate each keypoint with a descriptor, which is actually a 3D representation of the gradient distribution of the local region centred on the keypoint. The descriptor is highly distinctive and is robust to the remaining variations, such as changes in 3D viewpoint and illumination. The gradient-orientation histogram is used to describe a keypoint, which is generated from the gradient information of all pixels within the local region. Given a pixel, the gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are computed from Equation (2.12) and (2.13), respectively.

$$m(x, y) = \sqrt{G_x^2 + G_y^2} \tag{2.12}$$

$$\theta(x, y) = \tan^{-1}(\frac{G_y}{G_x}) \tag{2.13}$$

where G_x and G_y given below are the difference of smoothed pixel values in x and y directions, respectively.

$$G_x = L(x + 1, y) - L(x - 1, y) \tag{2.14}$$

$$G_y = L(x, y + 1) - L(x, y - 1) \tag{2.15}$$

As shown on the left of Figure 2-10, the length and direction of a particular arrow represents the gradient magnitude and orientation of each pixel, respectively.

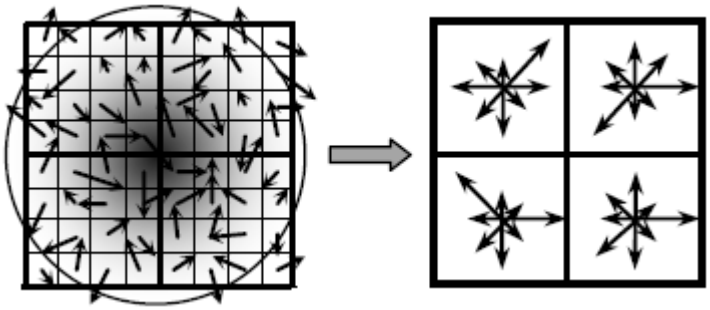


Figure 2-10: Descriptor generation for an 8×8 set of pixels.

To generate a descriptor, each local region around a keypoint is segmented into several square sub-regions, with each characterised by an 8-bin histogram. Figure 2-10 shows an example of a 2×2 descriptor vector computed from the local image patch of size 8×8 pixels, whereas 4×4 square sub-regions are used in the standard SIFT algorithm. Firstly, the image gradient and orientation are sampled around the keypoint, which are accumulated to generate a histogram summarising the contents of the entire local region. The orientation that corresponds to the bin with the largest magnitude in the histogram is the dominant direction of the local gradient distribution and is assigned to the keypoint, which is named as the principal orientation (θ_{po}). Secondly, the local region of size 16×16 pixels is segmented into 4×4 square sub-regions with each of size 4×4 pixels, and pixels within the local region are rotated with respect to θ_{po} for rotation invariance. Thirdly, each sub-region is characterised by an 8-bin histogram with each bin covering 45°. As shown on the right of Figure 2-10, each sub-region is described using an 8-bin histogram with the orientation of each bin representing 45°, and the length of the arrow represents the accumulated gradient magnitude of all pixels within the sub-region for each of the eight orientations. Finally, histograms of all sub-regions are linked together, leading to a final descriptor of 128 dimensions, as shown in Figure 2-11.

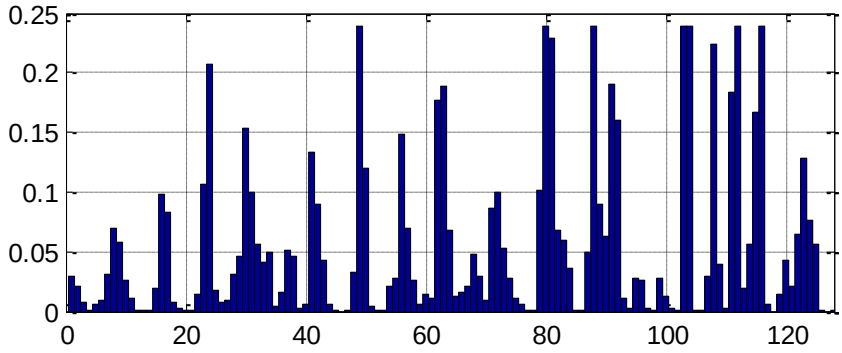


Figure 2-11: Final descriptor of 128 dimensions for standard SIFT algorithm.

2.3 SIFT Variations

Although the feature detection and description are often designed together, the solutions to these two problems are independently explored [1] [27]. In this section, an introduction is given to the algorithms that are explored as alternatives to SIFT with respect to detection and description, respectively.

2.3.1 PCA-SIFT

The PCA-SIFT [28] applied Principal Component Analysis (PCA) [29] to the standard SIFT algorithm for dimensionality reduction. Rather than using the orientation histogram to represent the gradient distribution within the local region, the PCA-SIFT applied PCA to the normalised gradient image patch centred on the keypoint. The inputs to the PCA-SIFT are identical to the standard SIFT, which are the keypoint location, scale, and principal orientation.

To generate a PCA-SIFT descriptor, a square patch is selected around a keypoint with size proportional to its scale value and the patch is then rotated relative to the principal orientation for rotation invariance. The gradient values in the patch are sampled such that for every keypoint the final patch is of size 41x41.By concatenating both the horizontal and vertical gradient maps for the 41x41 image patch, an input vector of size 39x39x2=3042 elements is created, which is normalised to reduce the effect of illumination changes. The final descriptor is of size n=20, which is generated by projecting the input vector into the feature space with dimensionality of n=20 using PCA.

The descriptor generation process takes comparable time for both PCA-SIFT and standard SIFT. The PCA-SIFT is more compact, leading to faster matching speed. However, according to the comparative study by Mikolajczyk [27], the PCA-SIFT is less distinctive than standard SIFT. Besides, the standard SIFT is better suited to handle errors introduced by orientation assignment and scale estimation [28].

2.3.2 Speeded Up Robust Features

Viola and Jones [30] proposed to use the integral image, which is also known as summed-area tables [31], in the context of real-time face detection. The entry of an integral image at location **x** is the sum of all pixels in the input image of a rectangular region formed by the origin and **x**. Given an integral image, it takes only four simple arithmetic operations to calculate the area of any sized rectangular region, as shown in Figure 2-12.

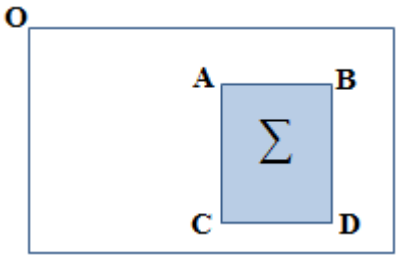


Figure 2-12: $\sum = I_{\Sigma}(D) - I_{\Sigma}(C) - I_{\Sigma}(B) + I_{\Sigma}(A)$.

H. Bay [32] [33] extended this idea further and proposed the Speeded-Up Robust Features (SURF), which makes use of the integral images that allows for box-type filters to approximate the determinant of Hessian matrix for fast feature detection. The idea of using box-type filter instead of Gaussian filter is that the Gaussian filter has to be quantised and cropped, and the approximation is pushed even further with box filters [33], as shown in Figure 2-13. The first two images are the quantised and cropped Gaussian second-order derivatives in y-direction (G_{yy}) and xy-direction (G_{xy}), respectively. The last two images are the box-filters (D_{yy}, D_{xy}) that approximates the Gaussian second-order derivatives in the first two images, respectively [33]. The Hessian matrix can be computed very fast using integral image and box-type filters, independent of the filter size. Interest points are localised by applying non-maximum suppression in a 3x3x3 neighbourhood, which are then refined in scale and image space using quadratic interpolation.

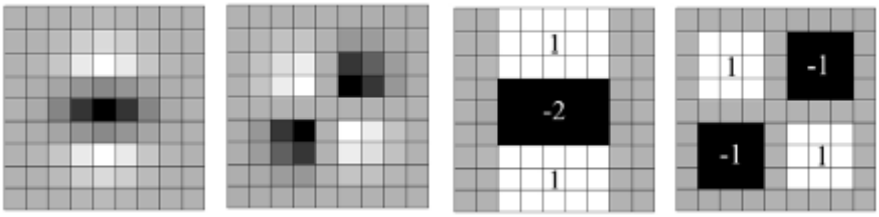


Figure 2-13: Box-filters.

The SURF descriptor is a histogram of the local distribution of Haar-wavelet [34] responses within the neighbourhood of the keypoint. Again, by exploiting the integral image, the Haar-wavelet response (d_x, d_y) in x or y direction can be computed within six operations at any scale. The local neighbourhood is split into 4x4 square sub-regions with each described by a four-dimensional descriptor vector $\mathbf{v} = (\Sigma d_x, \Sigma d_y, \Sigma|d_x|, \Sigma|d_y|)$for its underlying intensity structure, leading to a final descriptor of 64 dimensions. The sum of Haar-wavelet response in x and y direction can be split up according to the sign of d_x and d_y, respectively, thereby leading to a more distinctive representation of 128 dimensions. The SURF descriptor is computationally effective with respect to computing the descriptor's value at every pixel, but all gradients contribute equally to their respective bins, which results in damaging artifacts when used for dense computation [35].

2.3.3 Gradient Location and Orientation Histogram

GLOH [27], which is acronym of Gradient Location and Orientation Histogram, is considered as an extension to SIFT by using log-polar location grid. As shown in Figure 2-14, the local region is arranged with eight sub-regions in the angular direction and three sub-regions in the radial direction, resulting in 17 sub-regions. Mikolajczyk computes SIFT descriptor for each of the 17 sub-regions. With the gradient orientation quantised into 16 bins, the resulting histogram is of 272 bins, which is further reduced to 128 by applying PCA.

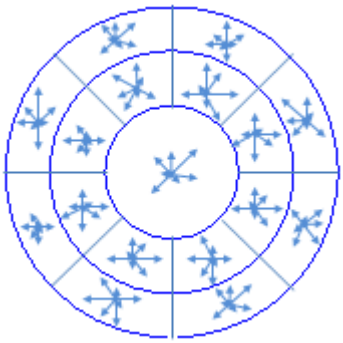


Figure 2-14: Spatial arrangement for GLOH descriptor with 17 sub-regions.

According to the performance comparison of different descriptors by Mikolajczyk [27], the GLOH descriptor obtains better results than SIFT in the presence of real geometric and photometric transformations. However, GLOH is more expensive to compute than SIFT.

2.3.4 DAISY

Inspired by the developments of SIFT and SURF, Tola [35] takes advantage of the log-polar grid with Gaussian weights from [36] and speeds up computation by applying Gaussian convolutions to orientation maps. The descriptor is named DAISY due to the flower like arrangement of the local region, as shown in Figure 2-15. The radius of each sub-region is proportional to the Gaussian kernels and the '+' sign represents the centre of each sub-region [35]. DAISY is a novel descriptor initially proposed for dense wide-baseline matching across stereo image pairs. It retains the robustness of SIFT and GLOH to perspective and lighting changes and can be computed quickly at every pixel. Unlike SURF, the DAISY descriptor can be computed efficiently at every pixel and does not introduce any artifacts that degrade the matching performance [35].

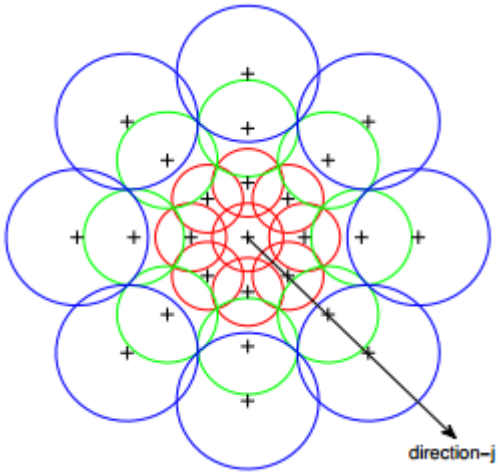


Figure 2-15: The DAISY descriptor with three rings of sub-regions in the log-polar spatial arrangement around the centre.

Figure 2-16 shows the construction process for Gaussian smoothed orientation maps with four discrete directions as an example. Four discrete orientations with each smoothed by three Gaussian kernels are used as an example to demonstrate the construction process. In practice, the DAISY descriptor quantises the gradient orientation to eight directions, resulting in eight gradient maps (G_{o_i}) with one for each orientation representing 45^o [35]. Each orientate map is then smoothed with Gaussian mask Σ_k, which results in a set of Gaussian smoothed orientation maps for each direction. The magnitude of the Gaussian smoothed orientation maps are the entries to the final descriptor. The DAISY descriptor is fast to compute in that the Gaussian smoothed orientation maps are computed instead of calculating the weighted sum as in SIFT, with which the descriptor generation process becomes simple indexing operation that uses the centre of each sub-regions as an index to the Gaussian smoothed orientation maps.

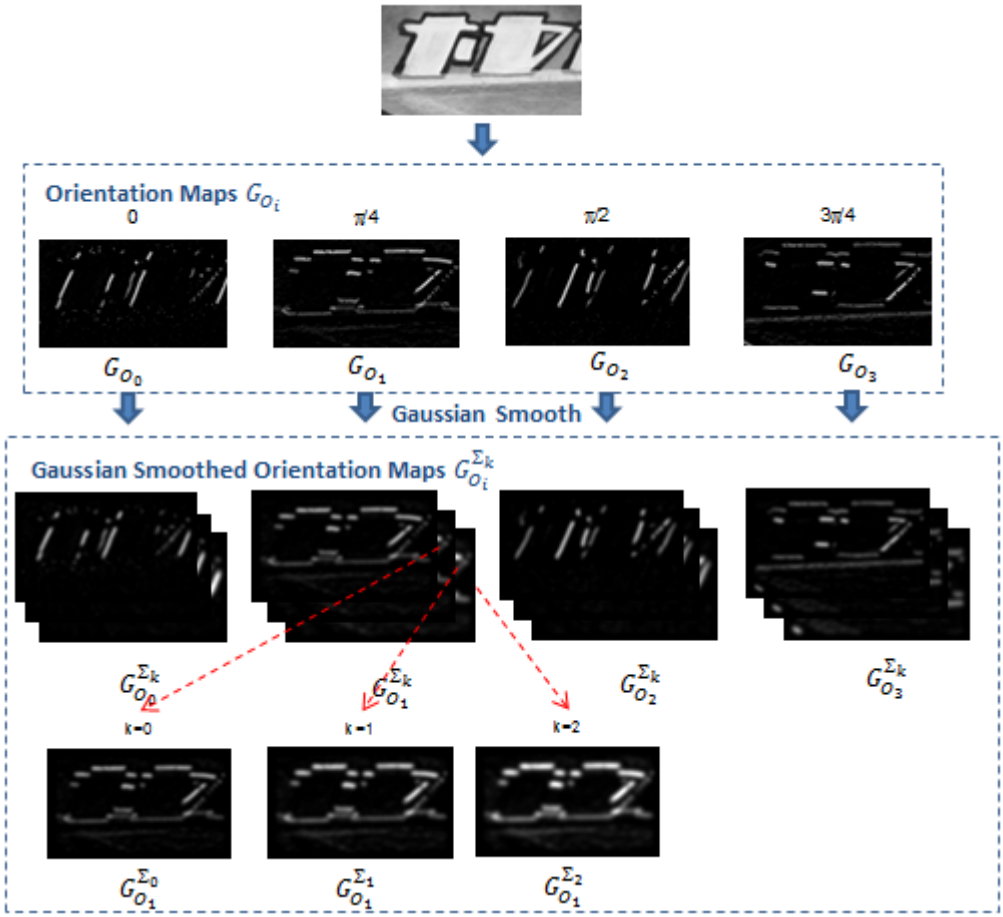


Figure 2-16: Construction of Gaussian smoothed orientation map for DAISY.

2.4 Hardware Designs

The existing researches aiming at accelerating SIFT using hardware is divided into three different categories: 1) optimising parallel algorithms based on multi-core processors [37] [38], 2) using state-of-art Graphics Processing Unites (GPUs) [39] [40] [41] to improve the processing efficiency, and 3) implementing SIFT using FPGA (Field Programmable Logic Array) by exploring the inherent parallel processing property of FPGA devices.

Numerous studies [42] [43] have compared the performance of FPGAs, GPUs and CPUs. Bodily [44] suggested that GPUs are not suitable for many embedded applications, such as intelligent robots with limited power supply, mainly because the

power consumption of GPUs is significant when compared to FPGA devices. In a most recent evaluation research [45], the performance and energy comparison of FPGAs, GPUs and multicores is conducted on a sliding-window applications due to their frequent usage in digital signal processing, such as sum of absolute distance and 2D convolution. They concluded that FPGA is generally faster than GPUs and multicores, and uses orders of magnitude less energy than other devices in many situations, providing the only realistic embedded system implementation for high-definition video. This section mainly focuses on the related FPGA designs for the SIFT algorithm, highlighting their advantages and disadvantages. The architecture proposed in each of them is analysed, as they are the most relevant publications to this project.

2.4.1 Hardware Design for Feature Detection

Se [46] implemented the SIFT detection on a Virtex II Xilinx FPGA to support a stereo vision system for robotic navigation. It takes 60 ms to extract SIFT features from VGA image and has achieved the performance improvement of 10 times in relation to a Pentium III 700 MHz processor. This is the first work reported in the literature in the field of SIFT extraction based on FPGAs, and marked the first attempt to accelerate SIFT using hardware. However, no architecture specifications have been provided. In [47], a partial implementation of the SIFT algorithm on FPGA is proposed to determine the translation and rotation between cameras for stereo vision. Only the Gaussian pyramid construction and keypoint detection is implemented in FPGA. The system is able to determine the verge angle between the two cameras with an accuracy of less than one degree with the system operating at 60 frames per second (fps) for input image of 640x240 pixels, which clearly shows the advantage of using FPGAs for solving intensive computer vision related tasks.

A hardware-software co-design is developed in [48], which partially implemented the SIFT algorithm on a Xilinx XUP-Virtex II Pro board. It takes only 0.8 ms to detect features from QVGA images with a clock frequency of 100 MHz. However, little information on the design architecture and FPGA resource usage has been provided. Bonato [5] proposed a detailed hardware architecture for vision Simultaneous Localisation And Mapping (SLAM) [49], which is able to detect features at up to 30 fps for QVGA images. As shown in Figure 2-17, the Gaussian

pyramid construction is divided by octave and Gaussian blurred images within each of the three octaves are computed in series, resulting in 18 Gaussian filter blocks.

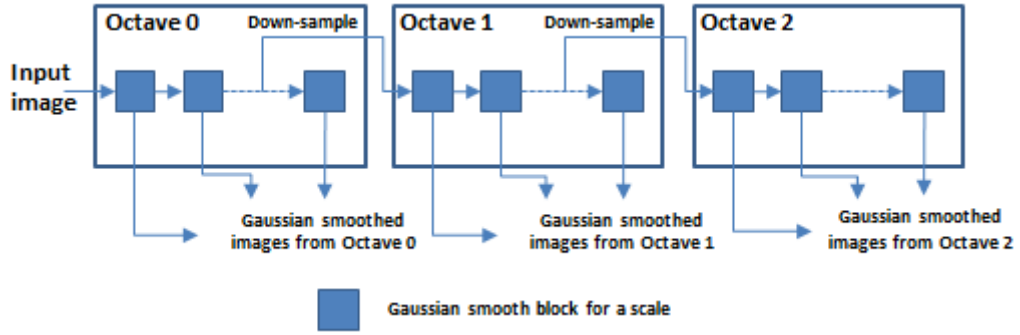


Figure 2-17: The architecture implementing the Gaussian filters cascade in [5].

In order to reduce the FPGA resources usage and speed up the design, 5-bit unsigned representation has been adopted for DoG (Different-of-Gaussian) images, with which local minima are ignored in the detection stage. Since the system performance may be degraded with many features ignored, the design may not be suitable for other general image processing applications, such as object recognition that requires a large number of features densely covering the target object.

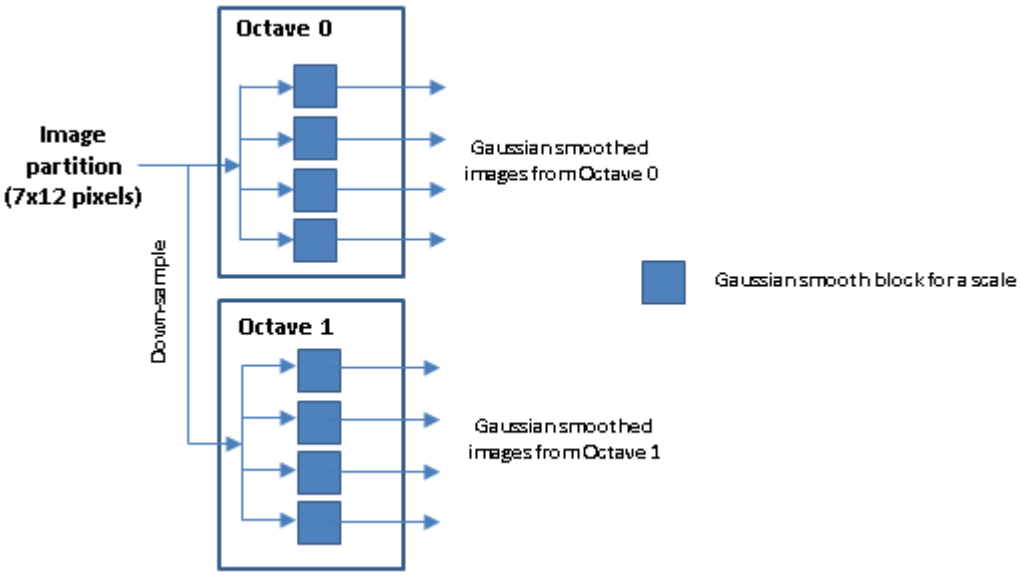


Figure 2-18: The fully parallel architecture for Gaussian filters in [50].

Yao [50] proposed a partition-based feature detection scheme, which is able to detect features from a VGA image within 31ms. To achieve real-time processing, the input image is segmented into partitions of size 7x12 pixels with 7 pixels processed in parallel. The Gaussian pyramid construction is simplified by using four smoothed images instead of six as suggested in the standard SIFT. The input to the second octave is generated by down-sampling the original image instead of the Gaussian smoothed image from the previous octave. Besides, the standard deviations of four scale images are set to 1.1, 1.3, 1.6 and 2.0, respectively. These simplifications reduce the total number of features and degrade the robustness to scale changes. Besides, the stability checking process for the keypoints is replaced by scaling down DoG pixels, which sacrifices the accuracy of features. Figure 2-18 shows the fully parallel architecture for Gaussian pyramid construction. It should be noted that each Gaussian smooth block consists of seven Gaussian filtering units working in parallel.

The overall processing time TC_{VGA} is defined below.

$$TC_{VGA} = \frac{(640-b)\times(480-b)+(320-b)\times(240-b)}{x\times y} \times [(x+b+2)\times(y+b+5)+2] \tag{2.16}$$

where b is the size of the boundary region caused by the nature of 2D Gaussian filter. x and y are the height and width of the image partition, respectively.

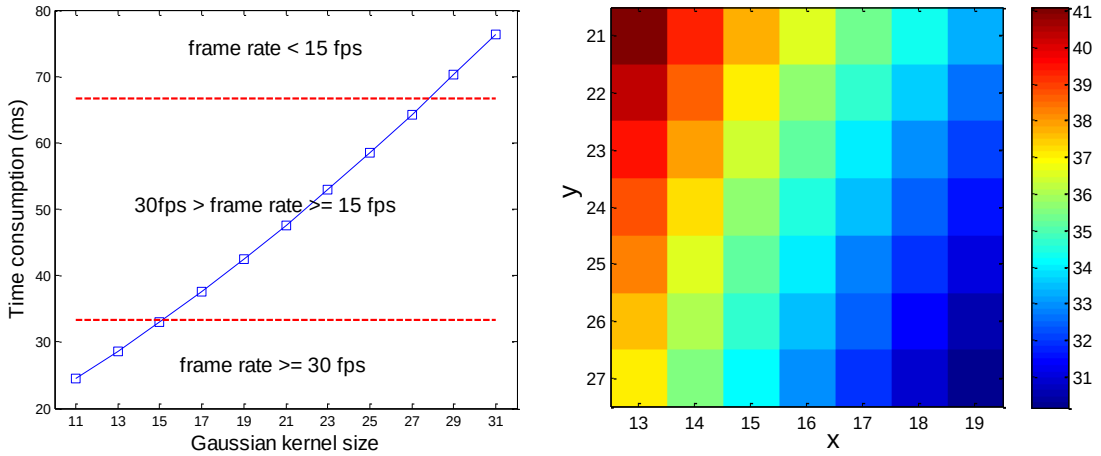
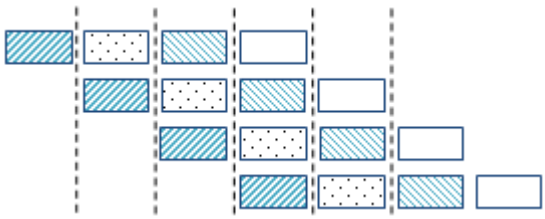


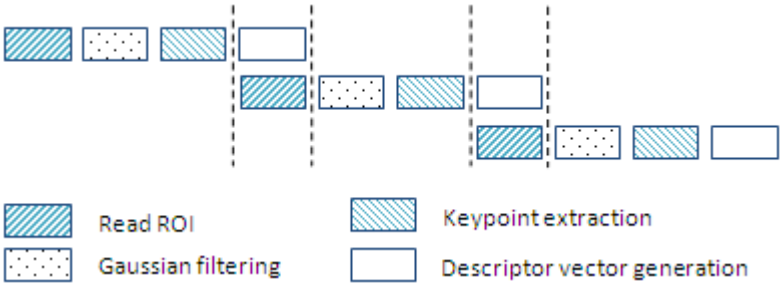
Figure 2-19: The left image shows the processing time for VGA image as a function of the Gaussian kernel size for [50]. The right image shows the processing time for XGA image as a function of the partition size.

This design is sensitive to both the size of Gaussian kernel and the input image. Figure 2-19(a) shows the processing time as a function of the Gaussian kernel. The issue of Gaussian kernel will be later addressed in Chapter 4. Figure 2-19(b) shows the processing time as a function of partition size for XGA (1024x768) images with Gaussian kernel of size 15, which shows that at least 16 pixels have to be processed concurrently to achieve real-time. This requires at least 16 Gaussian smooth units to be implemented in parallel. Increasing the size of either the Gaussian kernel or the input image will lead to a significant increase in the number of Gaussian smooth units, which is inefficient, in terms of hardware resource usage.

Another regions-of-interest (ROI) based scalable architecture is proposed in [51], which works in two different modes: high-speed mode and high-accuracy mode. As shown in Figure 2-20, the high-speed mode works in a pipelined architecture with ROI of size 40x30 pixels, while the high-accuracy mode works in a sequential architecture with ROI of size 80x60 pixels. The throughput of high-speed and high-accuracy mode is 56 fps and 32 fps, respectively, with a clock frequency of 50 MHz. The overall architecture for Gaussian smooth is similar to Figure 2-18, but each Gaussian smooth block consists of ten Gaussian filtering units working in parallel.



(a) Pipelined architecture in high-speed mode.



(b) Sequential architecture flow in high-accuracy mode.

Figure 2-20: Overall architecture for two different modes proposed in [51].

Kim [52] improved upon Bonato's work [5] to reduce the memory requirement by replacing the cascade Gaussian filter process with parallel processing and sharing Gaussian filter bank between octaves, as shown in Figure 2-21. By employing parallel architecture within each octave, the buffer storing the intermediate smoothing results in the cascade filtering mode is saved. The design is implemented on Altera Stratix II FPGA (EP2S60F672C3), and achieves a reduction in registers and LUTs of 58.6% and 36.6%, respectively. However, the overall throughput is not stated.

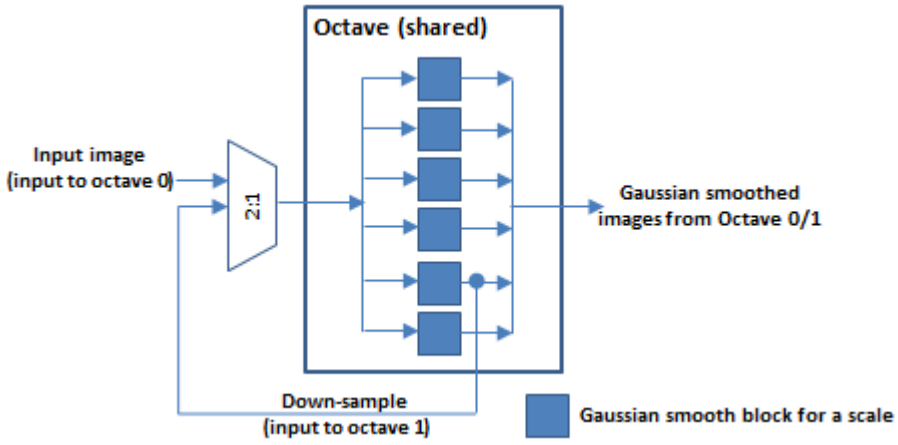


Figure 2-21: Overview of the parallel architecture with Gaussian filter bank shared between octaves in [52].

A SIFT hardware accelerator for real-time image feature extraction has been proposed by Huang [53]. The main contribution of this work is that the processing time for feature detection is reduced to 3.4 ms for VGA sized video by taking advantage of the image streaming method proposed in [54]. As shown in Figure 2-22, the design mainly consists of two interactive parts. Every time a feature is identified from the main processor, the co-processor is invoked to generate descriptor for the detected feature point. The main processor does not start detecting until the descriptor has been generated for the previously detected feature.

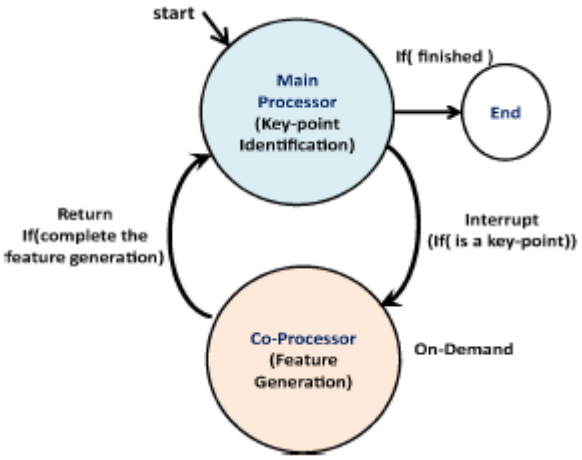


Figure 2-22: State transition diagram of two interactive components for [53].

The overall processing time TC_{VGA} is defined as follows.

$$TC_{VGA} = TC_{detection} + N_{features} \times TC_{description} \tag{2.17}$$

where $TC_{detection}$ is the time requirement for feature detection and is directly proportional to the size of the input image. $TC_{detection}$ is equal to 3.4 ms for VGA. $TC_{description}$ is the time requirement for generating a descriptor, which is equal to 33.1 us. $N_{features}$ is the number of descriptors to be generated.

Although feature detection has been significantly accelerated, the overall processing time is actually decided by the number of features as a result of the two interactive components working in series. Figure 2-23 shows the maximum number of features that can be processed in real-time for input images of different resolutions. The maximum number corresponds to an overall processing time of 33.3 ms. The number of features that can be processed within 33.3 ms decreases with the increase of image resolution, which indicates that the design is not applicable for processing higher resolution images that may produce larger number of features.

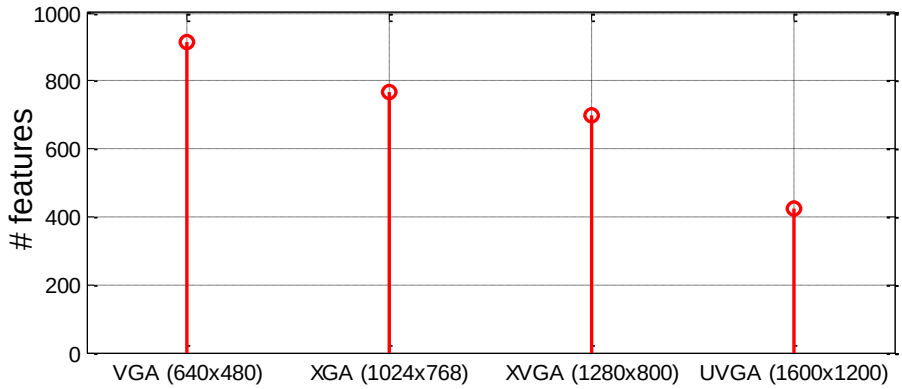


Figure 2-23: Maximum number of features for images of different resolution.

A most recent design that accelerates SIFT feature detection is proposed by Chang [55], which is improved upon their earlier work in [56]. Chang improved the processing speed by dividing the Gaussian pyramid construction process by scale, as shown in Figure 2-24. The design is able to detect features from QVGA images within 1.1 ms using Xilinx Virtex II Pro FPGA (XC2VP305FF-1152), which corresponds to 900 fps. The design introduces high control complexity as a result of the octave interleaving. Besides, the design covers only the local extrema detection from DoG space, whereas keypoint refinement process that contains complex matrix inversion computation is not addressed.

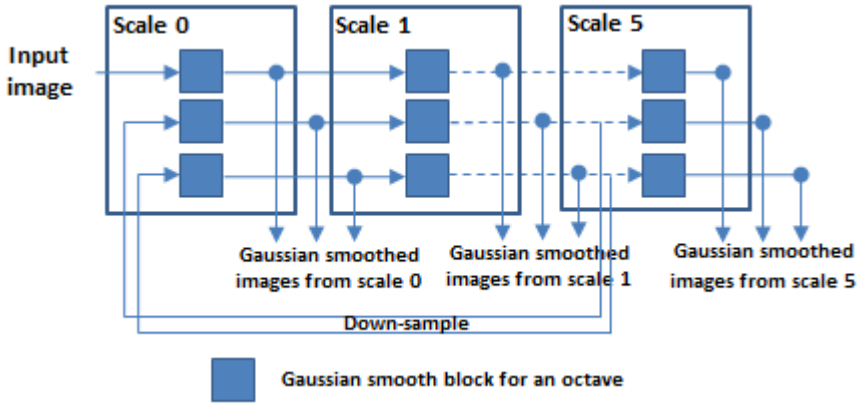


Figure 2-24: The architecture implementing the Gaussian filters cascade in [56].

2.4.2 Hardware Design for Feature Description

The above mentioned researches mainly focus on the FPGA implementation of feature detection. Considering that SIFT has the potential of detecting a large number of features and the time consumption of descriptor generation is proportional to the number of features detected, it becomes necessary to develop a high speed hardware architecture for descriptor generation that can be fully embedded on-a-chip for real-time applications.

Bonato [5] proposed parallel hardware architecture for feature detection, which is able to detect features at up to 30 fps for QVGA. However, the feature description is implemented using software, which requires 11.7 ms to generate a descriptor and has become the bottleneck that limit the overall throughput. Lin [57] proposed a VLSI architecture that takes 15.315 us to generate a descriptor, which corresponds to 65,300 descriptors per second with a clock frequency of 200 MHz. The design achieves 60 fps for VGA at approximately 1,088 features per frame. In a most recent design [53], a SIFT hardware accelerator for real-time feature extraction has been proposed. It takes approximately 33.1 us to generate a description. To achieve an overall throughput of 30 fps, the number of features is limited to 890 for VGA sized video. Besides, not much detail on the architecture of descriptor generation module has been provided in [53]. Considering the fact that SIFT has the potential of detecting a large number of features that densely covering the entire image, the number is likely to exceed 1,000 for a VGA image, so the throughput of [5] and [53] may not be large enough.

It can be seen from the review of the developed systems that the main drawback of the existing systems is the relatively low overall processing throughput with feature description incorporated. The drawback emerged mainly due to the computational complexity of the algorithm that gives rise to a large requirement in the processing time.

Chapter 3 The Optimised SIFT Algorithm

3.1 Introduction

In this chapter, an alternative to the spatial arrangement of the standard SIFT descriptor is proposed by taking advantage of the log-polar spatial arrangement of the DAISY descriptor. The standard DAISY is extended to be invariant to rotation and scale changes, which is termed as SRI-DAISY (Scale and Rotation Invariant DAISY). A novel keypoint matching strategy is also presented in this chapter, which provides better matching accuracy and higher hardware efficiency than the distance ratio based method from the SIFT.

3.2 Evaluation Criterion

The recall versus 1-precision curve has become popular evaluation criterion that is widely used in the context of matching and recognition. Given two images of the same object or scene, the recall is defined as the ratio of the number of correctly matched feature points to the number of correspondences. The precision is the ratio of the number of correct matches to the total number of matches.

$$recall = \frac{\#\ correct\ matches}{\#\ correspondences} \tag{3.1}$$

$$precision = \frac{\#\ correct\ matches}{\#\ total\ matches} \tag{3.2}$$

The correspondences are regarded as potential features that can be correctly matched between the pair of images with transformation.

The F-measure, which considers both recall and precision, reaches its best value at 1 and worst score at 0.

$$F_\beta = \frac{(1+\beta^2)\cdot precision \cdot recall}{\beta^2 \cdot (precision + recall)} \tag{3.3}$$

where β is the parameter that controls the balance between recall and precision. When $\beta = 1$, F_β becomes the harmonic mean of recall and precision. If $\beta > 1$, F_β puts more emphasis on recall. If $\beta < 1$, F_β becomes precision-oriented. In the evaluation results presented in this thesis, β is set to 1, giving equal emphasis on recall and precision.

3.3 Problem Analysis

It has been reviewed in Chapter 2 that currently existing designs are generally focused on investigating the parallelism of feature detection module to fully embed this part on a chip, whereas not much effort has been placed on improving the throughput of description generation. Actually, descriptor generation has become the bottleneck of the overall system due to both the high dimension of descriptors and the huge time requirement to process a large number of features. According to the literature review presented in Chapter 2, researches focusing on improving the efficiency of feature description fall into two categories:

- Exploration of descriptors that are more robust, with less computational complexity and are much faster to be evaluated.
- Development of efficient hardware architecture by exploring the parallel processing property of descriptor generation.

The performance of several widely applied descriptors has been evaluated in [27], which shows that the circular arrangement has better localisation properties than the grid layout of SIFT. Winder [36] [58] performed more extensive evaluation into different layout of descriptors and showed that DAISY [59] outperforms SIFT. GLOH [27] is the most robust descriptor among all kinds of proposed descriptors but with high computational complexity. SURF [32] [33] is a widely accepted algorithm that offers the fastest speed at the cost of higher memory consumption while the performance is not quite satisfying.

DAISY is faster to compute than SIFT, but the descriptor is initially proposed for dense wide-baseline matching and does not deal with rotation and scale changes.

Fischer [60] implemented Tola's [59] DAISY descriptor for fast computation, which explored the rotation invariance of the standard DAISY. The rotation invariant DAISY is termed O-DAISY, which is generated by rotating the descriptor relative to the principal orientation ($\theta_{po)}$ in a similar way to SIFT. The design is able to process images of 2034x2048 pixels at 30 fps, or VGA images at 406 fps, making it an optimum solution for dense wide-baseline matching. However, O-DAISY suffers from the following major drawbacks, which make it not suitable for general matching tasks with large geometric transformations.

- The orientation map of the standard DAISY descriptor is a quantised version of SIFT's orientation, and hence the rotation invariance is degraded when compared to SIFT as a result of the reduced precision of θ_{po}. In SIFT, θ_{po} corresponds to the direction of the largest bin in the 36-bin histogram generated based on the gradient distribution of the local region, where each bin represents 10^{o}. In the standard DAISY, orientation maps of eight discrete directions are computed with each representing 45^{o}, which limits the orientation up to eight discrete directions. Although Fischer increased the number of orientations from 8 to 16 to improve the precision of θ_{po} at the cost of doubling the computational complexity, the precision of θ_{po} is still limited to 16 for O-DAISY with each representing 22.5^{o}, which potentially degrades the rotation invariance of the descriptor.
- The distinctiveness of descriptors is reduced as a result of the spatial information discarded. Orientation maps of all 16 directions are involved in the computation of θ_{po}. However, only every other orientation map is involved in descriptor generation so as to avoid the increase in descriptor dimension.
- Gaussian smoothed orientation maps of each direction have to be buffered, resulting in a huge memory requirement. The memory requirement is directly proportional to the resolution of input images, the number of discrete orientations, the number of Gaussian smoothed orientation maps for each direction, and the precision of the gradient magnitude of each pixel. The number of Gaussian smoothed orientation maps correspond to the number of rings of sub-regions in the log-polar spatial arrangement around the centre.

- The scale invariance is not addressed in O-DAISY, resulting descriptors sensitive to scale changes.

Inspired by these evaluation results and hardware design, the SIFT detection is integrated with the SRI-DAISY, which is a DAISY-like local region arrangement that is adaptive to rotation and scale changes. The SRI-DAISY is faster to compute without performance degradation when compared with the standard SIFT descriptor, and is more robust to image rotation and scaling when compared with O-DAISY.

3.4 SRI-DAISY

In this section, the parameters that affect the spatial layout of the SRI-DAISY are studied. In general, the throughput of descriptor generation module is proportional to the number of keypoints to be described. The key factors that affect the processing time of a descriptor are the sub-region arrangement, the size of local region, and the throughput of memory interface for data access, such as GMOs. The memory interface will be discussed in Chapter 5.

3.4.1 Spatial Arrangement for SRI-DAISY

There are two parameters to be considered for the overall layout of the SRI-DAISY descriptor: the number of rings and the number of sub-regions on each ring.

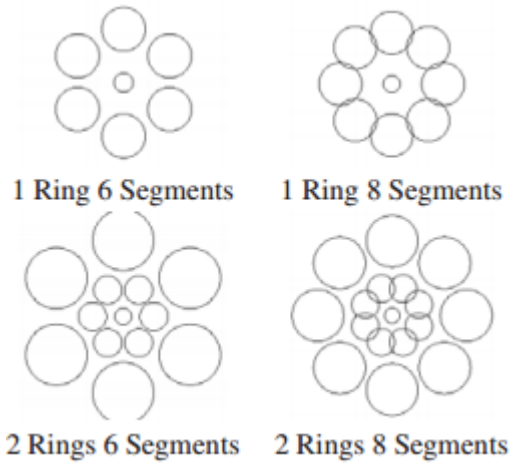


Figure 3-1: Typical spatial arrangement for DAISY descriptor studied in [58].

Typical spatial arrangements shown in Figure 3-1 have been studied by Winder [58], which shows that arrangement with two DAISY rings gives lower error rates than that with a single ring, and 8 sub-regions per ring performs better than 6 sub-regions per ring, as shown in Table 3-1. Besides, the error rate falls significantly when the number of discrete orientation is increased from 4 to 8, after which it shows little improvement. So 8-bin histogram for each sub-region is a proper choice, which is consistent with the standard SIFT.

Table 3-1: Error rates for different arrangement of local region [58].

Number of discrete orientations (per sub-region)	**1 Ring**		**2 Rings**	
	6	8	6	8
4	34.43	34.24	29.05	28.64
8	27.89	26.52	23.28	22.94
12	26.55	26.19	22.85	22.57
16	26.93	26.28	22.59	22.75

Apart from the error rate, the dimension of the descriptors is also very important, because it increases the computational complexity of both the descriptor generation and matching process. Besides, higher dimension also increases the memory requirement for buffering the descriptors. The descriptor dimension for different arrangement is given in Table 3-2.

Table 3-2: Descriptor dimension for different arrangement of local region.

Number of discrete orientations (per sub-region)	**1 Ring**		**2 Rings**	
	6	8	6	8
4	28	36	52	68
8	56	72	104	136
12	84	108	156	204
16	112	144	208	272

With a trade-off made between performance and hardware efficiency, arrangement with 1 ring 8 sub-regions each is used in this design, which results in the final descriptor of 72 dimensions.

3.4.2 Parameters for SRI-DAISY

The SRI-DAISY descriptor is generated from the local region, which size is proportional to the detection scale of the keypoint. Each circular sub-region is smoothed by a Gaussian kernel with standard deviation proportional to the detection scale of the keypoint. Typically, larger sub-region contains more information and hence is more distinctive to survive large transformations. However, it stands a higher chance of being occluded. Besides, the computational complexity of Gaussian smooth is closely related to the kernel size applied to each sub-region, so larger region results in higher computation workload. Therefore, local region arrangement has to be decided with a trade-off made between performance and computation efficiency.

Figure 3-2 shows a typical SRI-DAISY descriptor that is arranged with one ring in the radial direction of eight surrounding sub-regions on the ring. Each circle represents a sub-region. The '+' sign in the centre of the local region is the keypoint.

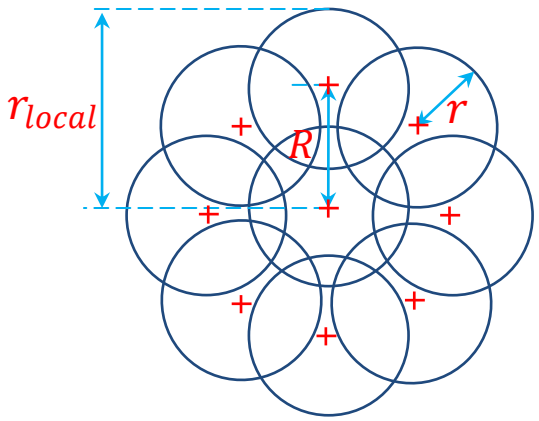


Figure 3-2: The SRI-DAISY descriptor arrangement.

In Figure 3-2, R is the distance from the keypoint to the centre of sub-regions, r is the radius of the sub-regions, and r_{local} is the radius of the local region for descriptor generation. The relationship between R, r and r_{local} is given in Equation (3.4), which shows that for a given local region size r_{local}, if one of the two parameters r and R is fixed, the other one is known.

$$r_{local} = R + r \tag{3.4}$$

In this section, three parameters are studied for SRI-DAISY descriptor, which are closely related to the spatial layout of the descriptor:

1. Standard deviation (σ_{DAISY}) of the Gaussian kernel that is applied to the sub-regions. σ_{DAISY} is proportional to r_{local}, and the ratio ($Ratio_{\sigma_{DAISY}}$) of σ_{DAISY} to r_{local} is studied.
2. Sub-region radius (r). The ratio of r to r_{local} is studied.
3. Region size factor F_{region}, which is the ratio of the diameter of the local region to the detection region, and $r_{local} = F_{region} * 3\sigma$.

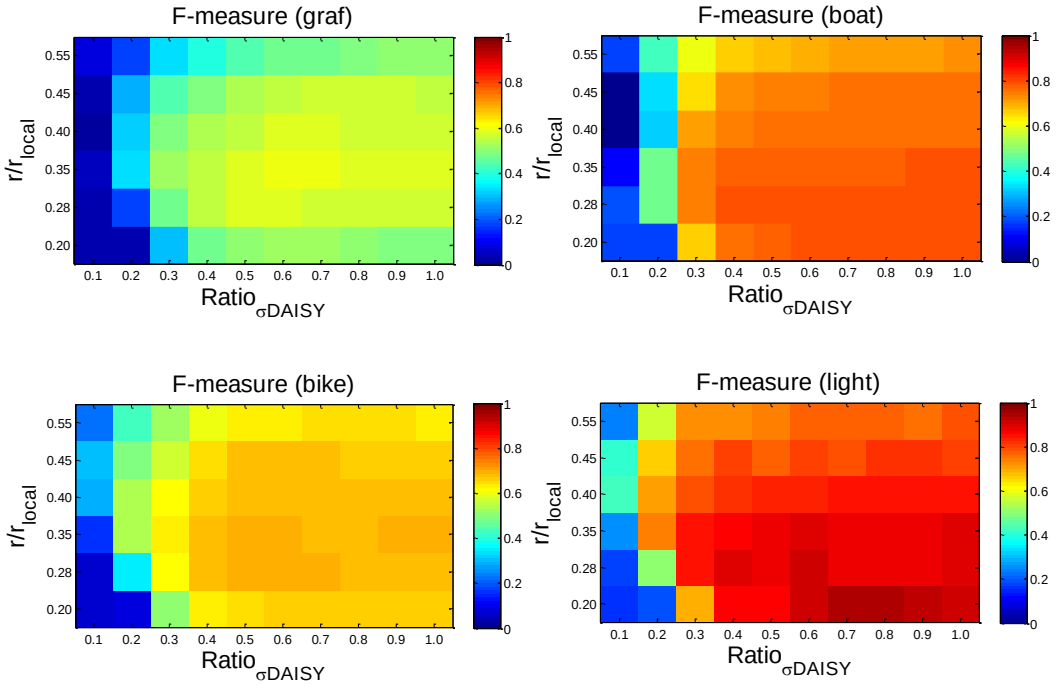
A distinction is made between the detection region and the local region prior to evaluating the effect of different parameters. The detection region is a collection of pixels that have effectively contributed to the SIFT detector response, whereas the local region is the region on which the descriptors are generated.

A wide range of settings has been studied, and some example results are given in Figure 3-3. The testing images are obtained from the website of Krystian Mikolajczyk [61]. These images are captured specifically aiming to test and compare keypoint detectors and local descriptors. Figure 3-3(b) shows that there does not exist one set of parameters that achieves the best performance for all types of images and transformations.

Each parameter is studied in detail in the following sub-sections. Firstly, the matching performance is checked as a function of the Gaussian kernel applied to each sub-region. Secondly, experiments are conducted to see how the matching performance varies with the changing of the sub-region radius, which decide the

spatial layout of the DAISY descriptor. Finally, the impact of region size factor F_{region} is evaluated.

(a) Example images. The first row shows reference images. The second row shows the transformed images. From left to right: graf (viewpoint), boat (scaling+rotation), bike (blur), and light changes.



(b) F-measure as a function of $^{r}/_{r_{local}}$ and $Ratio_{\sigma_{DAISY}}$.

Figure 3-3: Matching performance as a function of different parameter settings for SRI-DAISY descriptor arrangement.

a. Gaussian kernel

Figure 3-4 shows the matching performance as a function of $Ratio_{\sigma_{DAISY}}$, which is collected from a database of images with a wide range of transformations. . The x-axis is the index to the $Ratio_{\sigma_{DAISY}}$ in range [0.1, 0.5] of interval 0.05. In general, the matching performance improves as $Ratio_{\sigma_{DAISY}}$ increases. The performance becomes rather stable at around 0.35 and shows little improvement beyond that point.

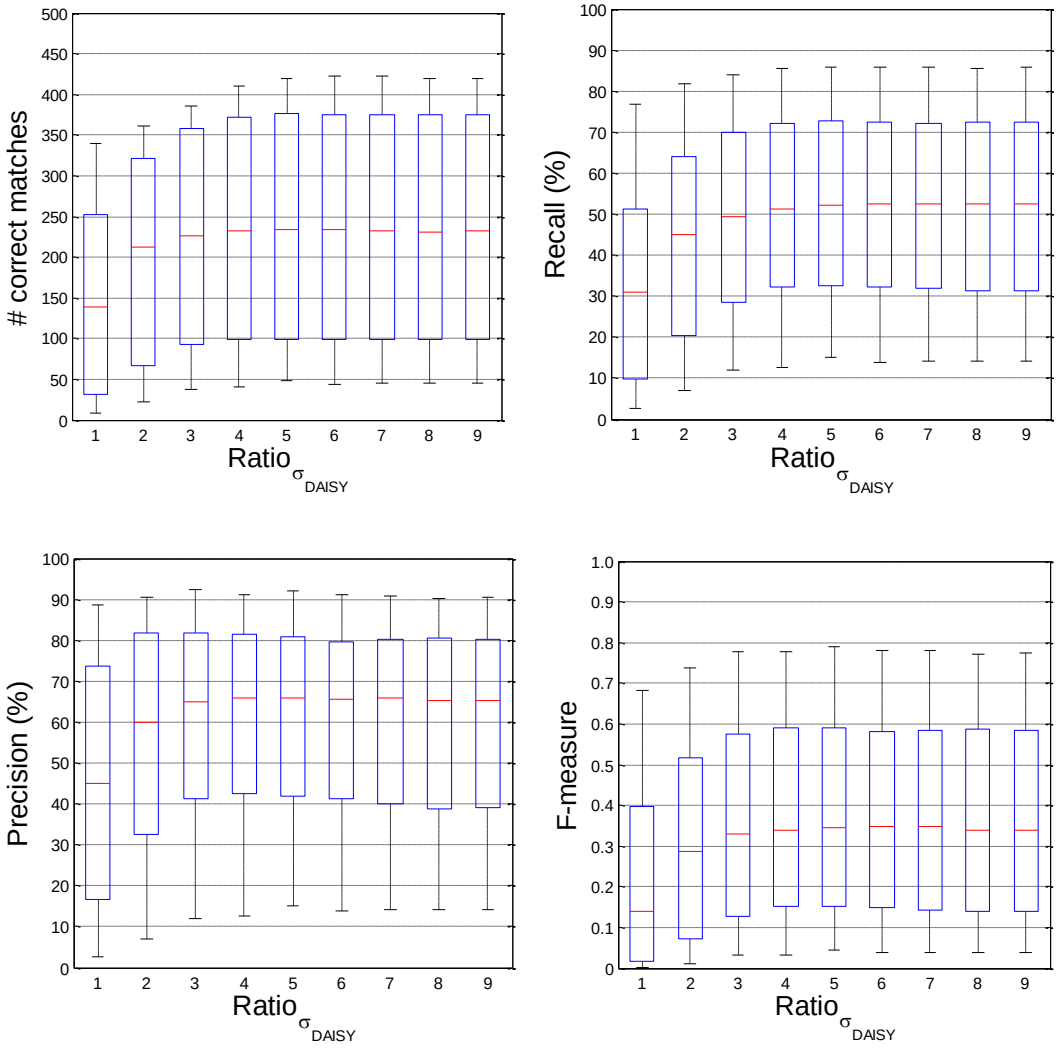


Figure 3-4: Matching performance as a function of the ratio between σ_{DAISY} and r_{local}.

b. Sub-region Radius

Experimental results are discussed to see how the matching performance varies with R and r for a given local region size r_{local}. The local region is sampled with a radius of four times the detection region (F_{region}=4.0). Figure 3-5 shows the example spatial layout of the local region. The ratio r/r_{local} is gradually increased, and hence the sub-region radius varies.

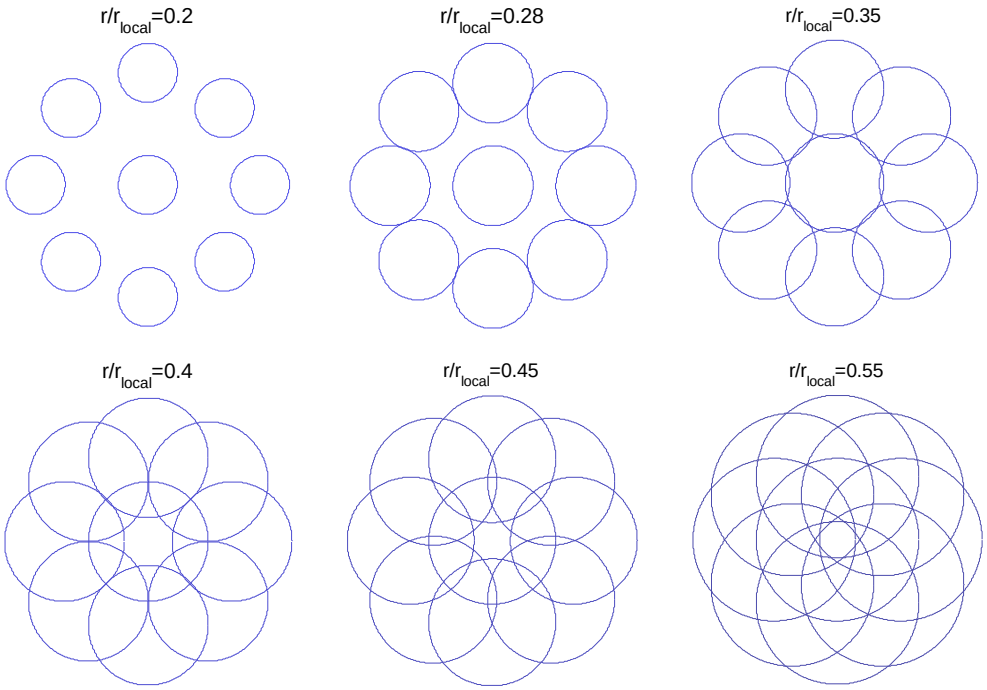


Figure 3-5: Local region arrangement for DAISY descriptor with one centre sub-region plus a ring of eight sub-regions.

The overall matching performance of different arrangement for SRI-DAISY is shown in Figure 3-6. In general, the performance is at a similar level for the selected descriptor arrangements, and r/r_{local} =0.35 is slightly superior to the others. Individual experiments are further conducted to see the effect of sub-region arrangement on different transformations.

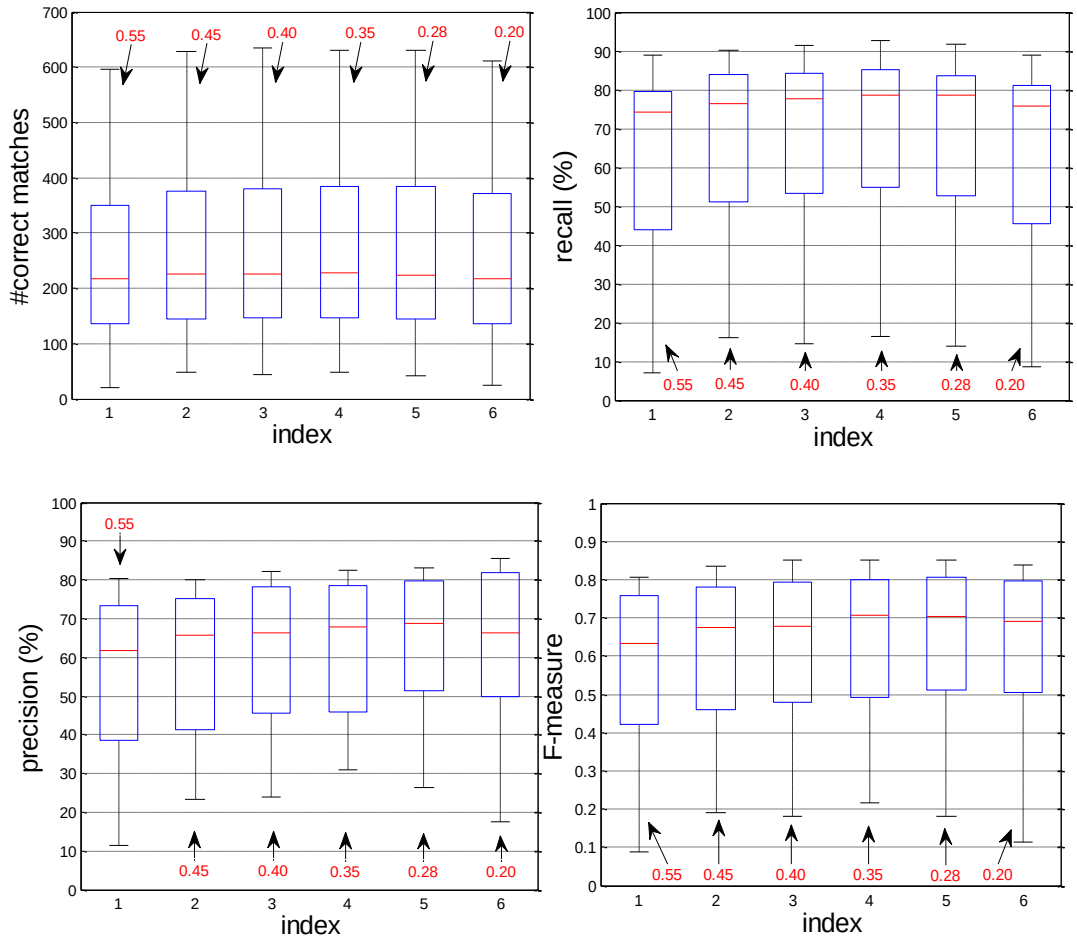
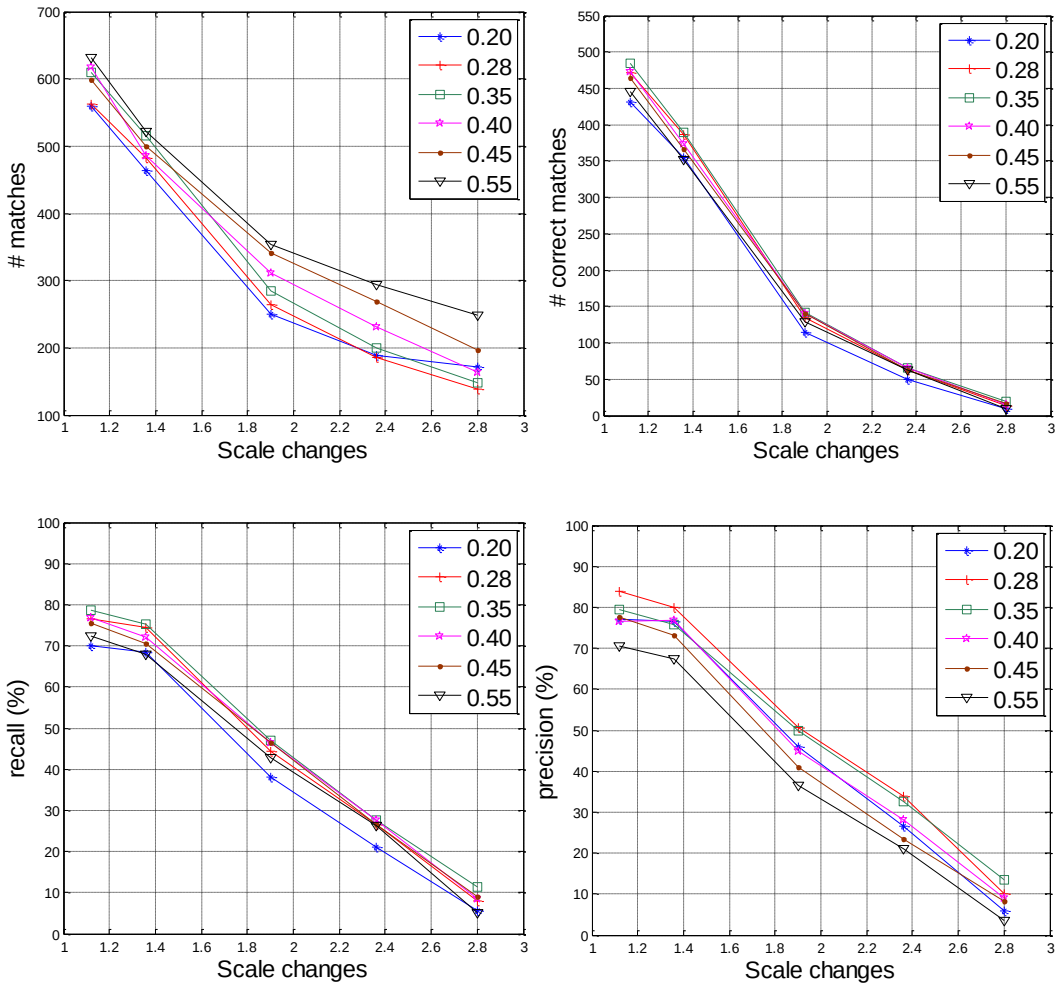


Figure 3-6: Matching performance as a function of r/r_{local}.

Figure 3-7(b) shows the matching results for a set of structured scene (boat) with in-plane rotation and scale changes shown in Figure 3-7(a). The structured scene contains distinctive edges with homogeneous regions. The recall and precision are virtually the same for r/r_{local}=0.28 and 0.35, and are slightly superior to the others. Similar observation has been made for the textured scene (wall), as shown in Figure 3-8. The textured scene consists of repeat textures. The precision of r/r_{local}=0.20 for textured scene holds a similar value to that of r/r_{local}=0.28 and 0.35, which is mainly due to the relatively smaller number of total matches when compared with r/r_{local}=0.28 and 0.35.

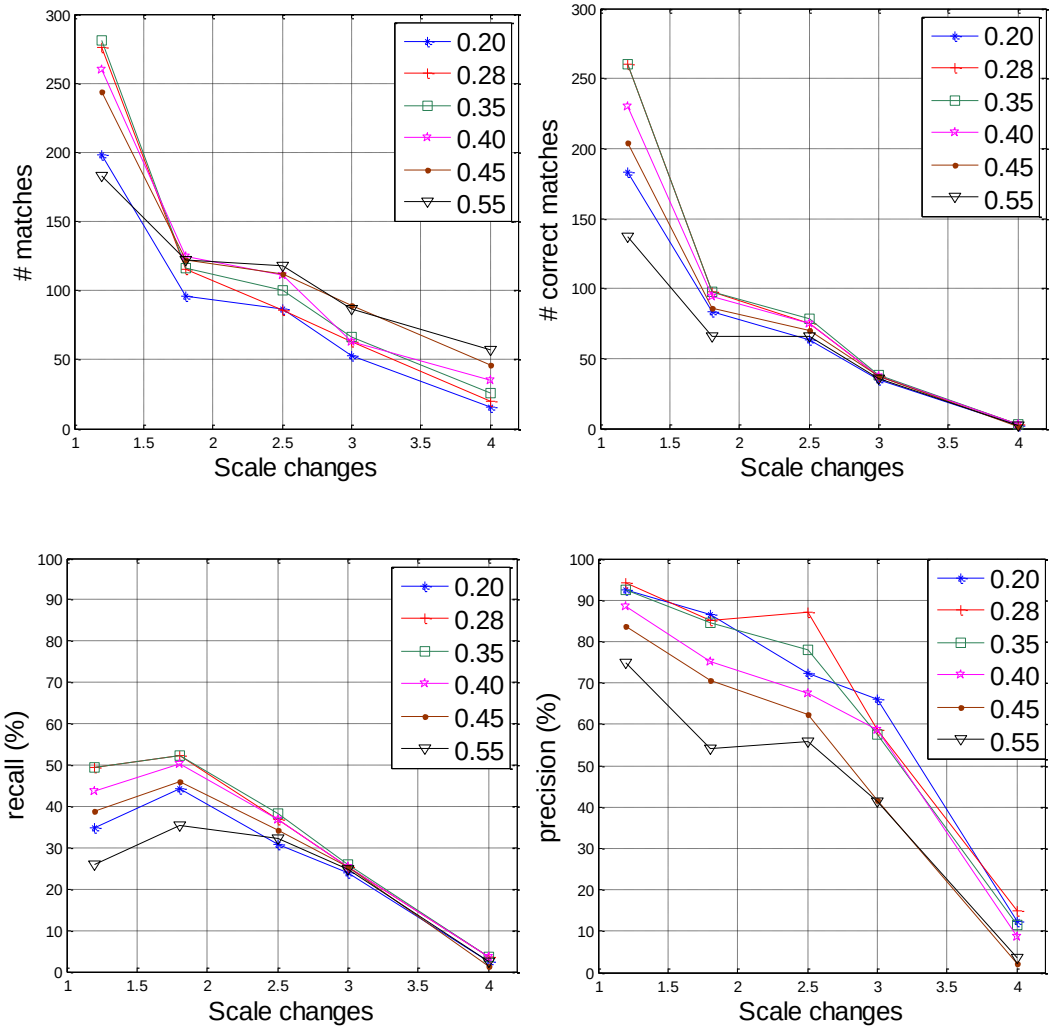
(a) boat



(b) Matching results

Figure 3-7: Matching results as a function of r/r_{local} for the boat set.

(a) Textured scene (wall)



(b) Matching results

Figure 3-8: Matching results as a function of r/r_{local} for the wall set.

The overall performance is reflected in the curve of F-measure, as shown in Figure 3-9. The best accuracy is achieved by r/r_{local} set to around 0.3 for both structured scene and textured scene. And the effect of spatial arrangement is more apparent for textured scene.

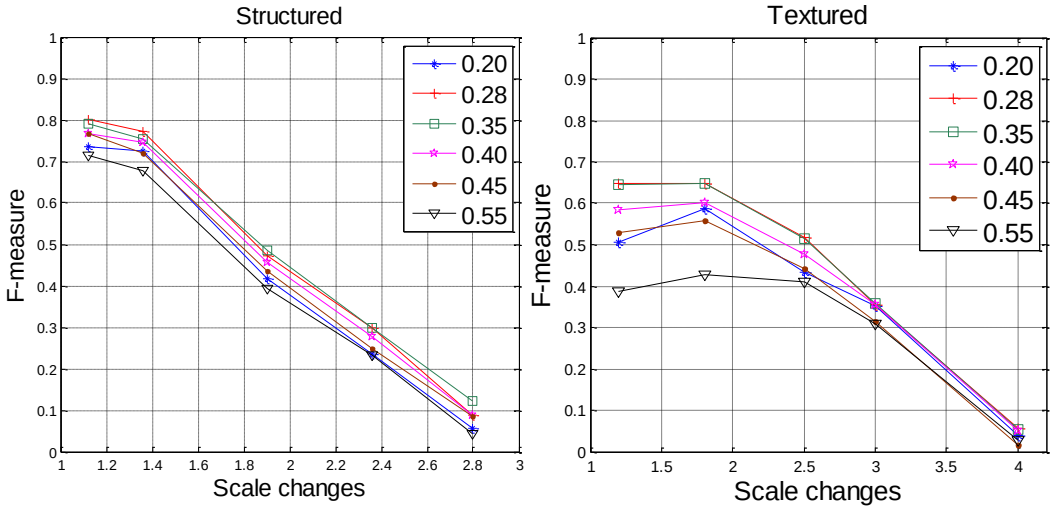


Figure 3-9: F-measure as a function of r/r_{local}. The left image shows the F-measure for the boat set. The right image is for the wall set.

Figure 3-10 to Figure 3-12 shows the matching performance as a function of r/r_{local} for images with transformation of viewpoint angle, image blur and illumination, respectively. The performance of radius ratio r/r_{local}=0.35 is superior to the others in most cases and is chosen to parameterise the design.

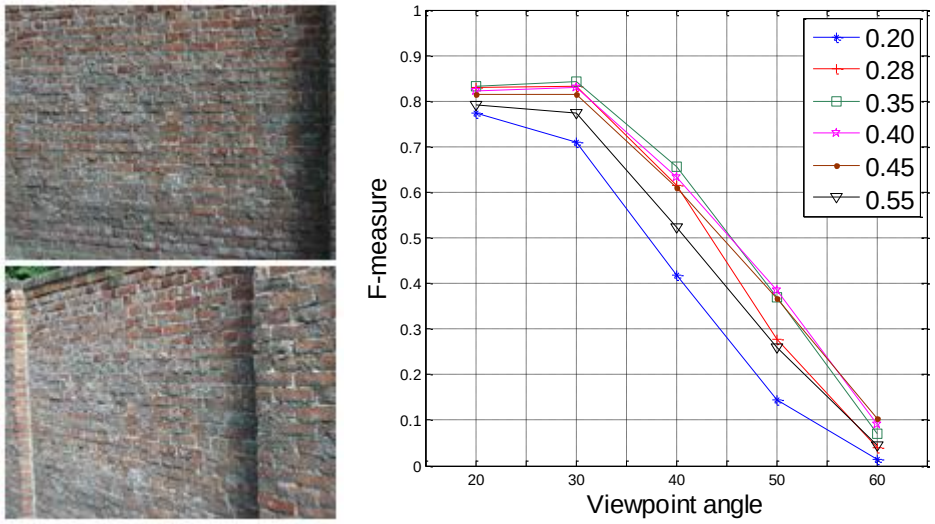


Figure 3-10: Matching result for textured scene (wall) with viewpoint angle.

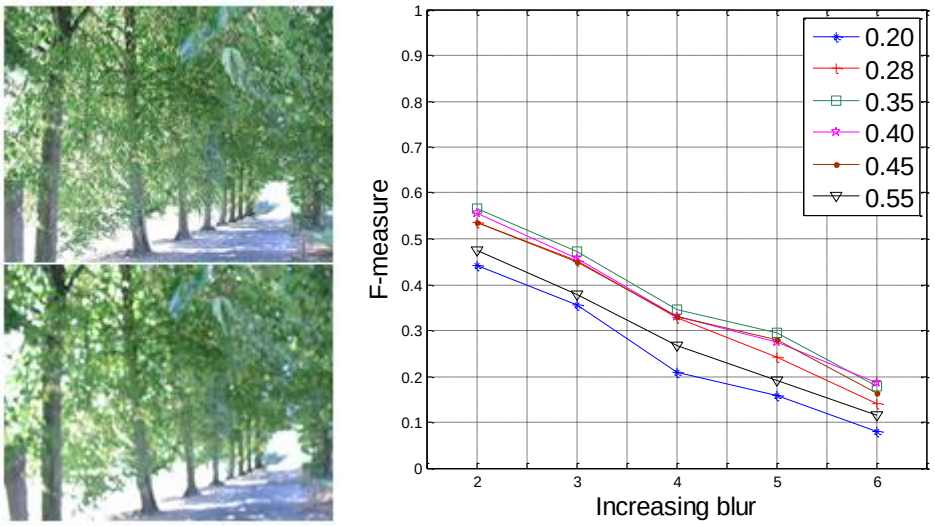


Figure 3-11: Matching result for textured scene (tree) with image blur.

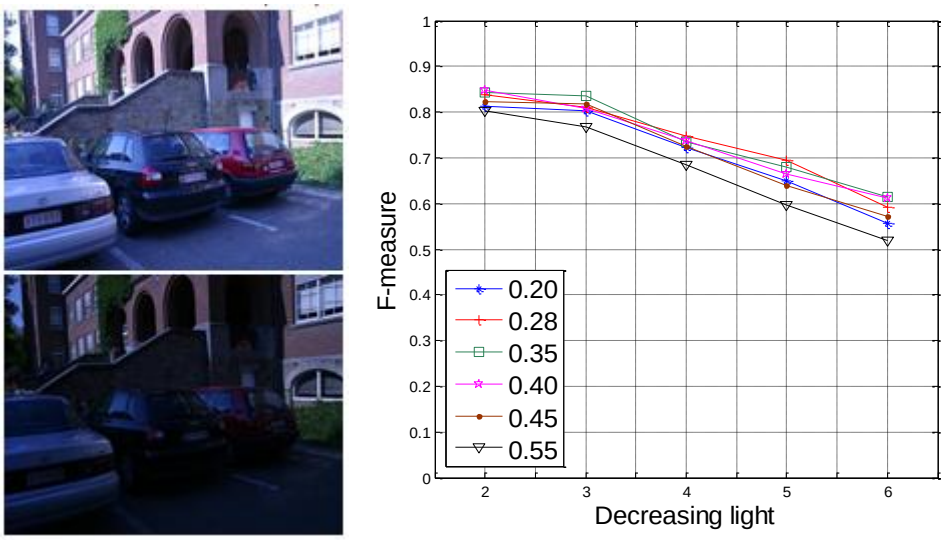


Figure 3-12: Matching result for images with illumination changes.

The above mentioned experimental results shows that the robustness of the descriptor can be improved by increasing the overlapped region, but only up to a certain point, after which the robustness drops. Therefore, r/r_{local} is set to 0.35. The final spatial arrangement is shown in Figure 3-13.

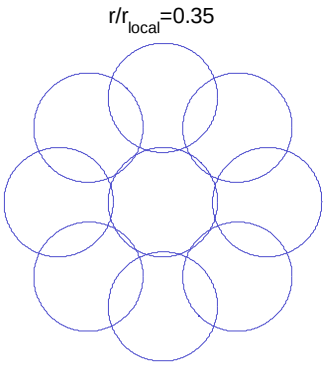


Figure 3-13: Determined spatial arrangement for DAISY descriptor.

c. Region Size Factor

Experiments are conducted to check the effect of region size factor F_{region} to see how the matching performance varies for different F_{region} for a given spatial arrangement. A descriptor is actually a 3D representation of the gradient distribution of the local region centred on a keypoint. Figure 3-14 shows how the matching performance varies for different F_{region} on a database of images covering a wide range of scene types and transformations. In general, the overall matching performance improves with the size of local region.

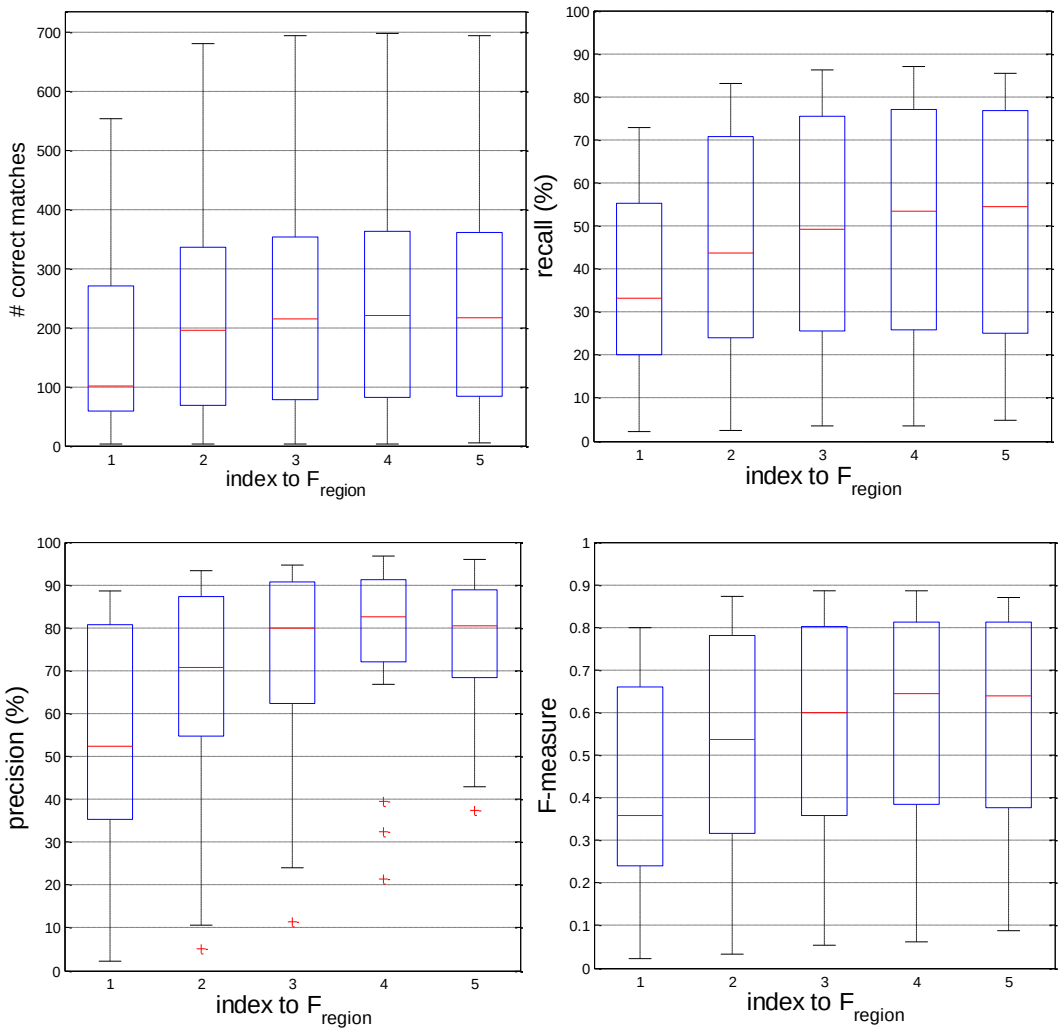


Figure 3-14: Matching results for F_{region} in range [2.0, 6.0] from a database of images with different scene type and transformation.

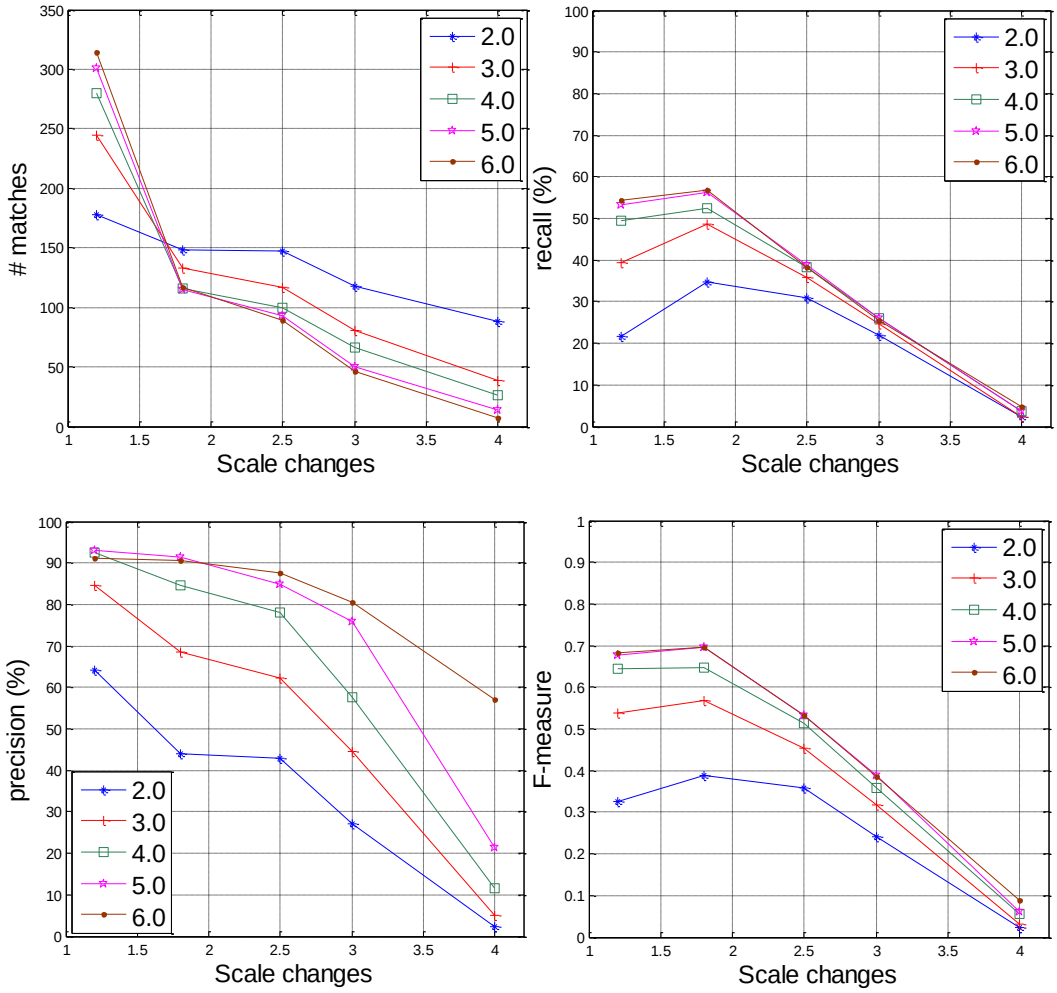


Figure 3-15: Matching results for F_{region} on the wall set.

Detailed experimental results for the wall set are given in Figure 3-15. The ranking for the number of matches is reversed when the scaling factor is larger than 1.5, but the ranking for the number of correct matches does not follow that of the total matches, which is reflected in the recall. This is because the descriptors are less distinctive for smaller regions and the distance between descriptors are on the average smaller, which leads to many incorrect matches and hence a lower precision. The slope of precision curve reflects the invariance of the descriptor to the transformation. The precision drops faster for smaller regions than larger ones, reflecting that small regions do not contain enough information to be correctly matched under large image transformations and larger regions are rather stable for large transformations. Larger regions typically contain more information and hence

the corresponding descriptors are more distinctive, making them easier to be corrected matched under large transformations.

However, larger regions stand a higher chance of being occluded and the cost of processing larger regions are higher, in terms of both hardware resource usage and processing time. Most of the time devoted to descriptor computation is spent on convolutions, and the computation workload of convolution is directly proportional to the size of the sub-regions that increases linearly with F_{region}, as shown in Figure 3-16. As a result, larger F_{region} leads to a significant increase in the computational workload and also the processing time.

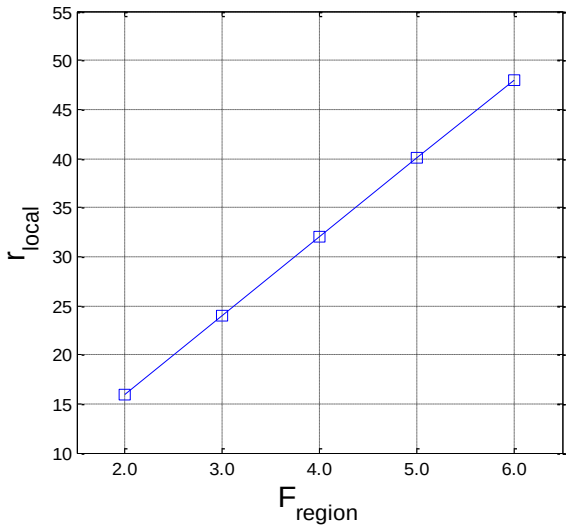


Figure 3-16: Local region size as a function of F_{region} for descriptors.

Experiment results for other types of transformation are given in Figure 3-17, which shows how the matching performance of a given type of image transformation varies with F_{region}. The experimental results of F_{region} beyond 3.0 are at a similar level.

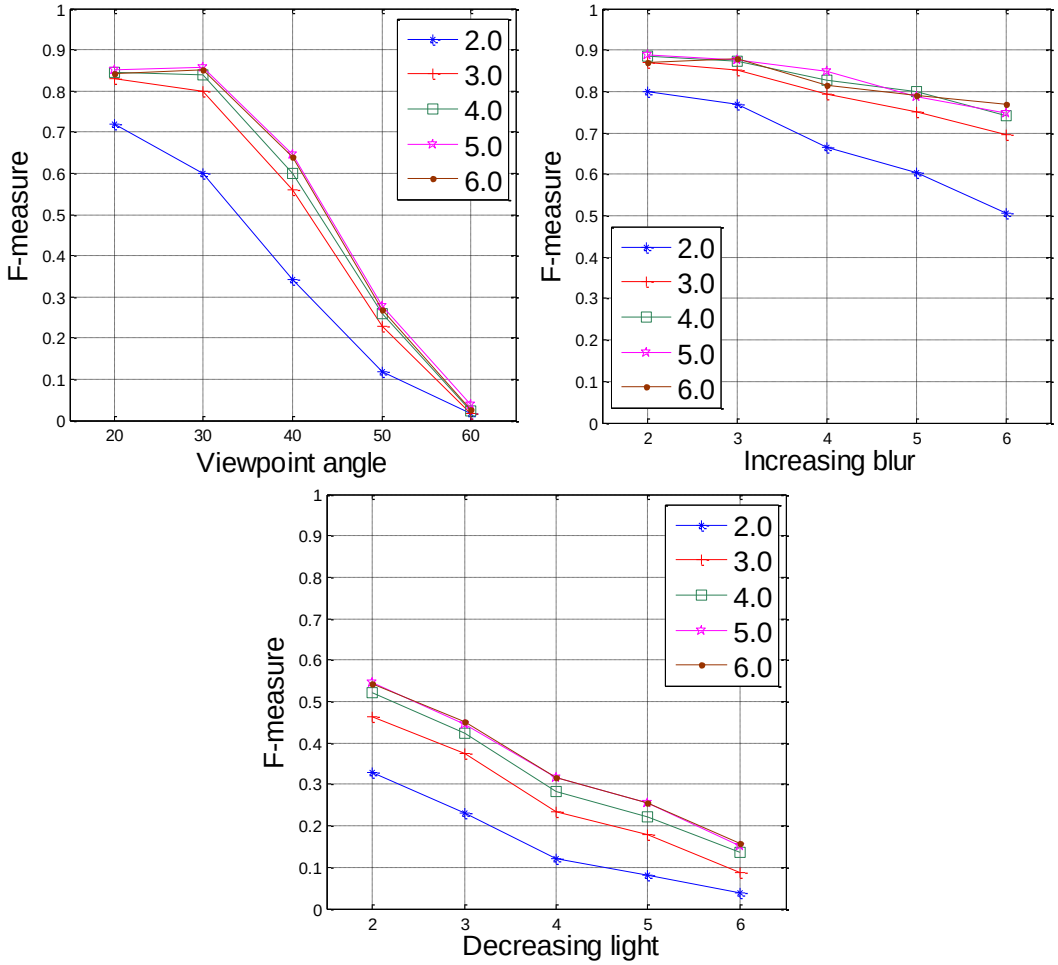


Figure 3-17: Matching results for different region size factors for sets of images with viewpoint changes, image blur and illumination.

As a result of the above experiments, the spatial layout of SRI-DAISY descriptor is arranged with one centre sub-region plus a ring of eight surrounding sub-region. With each sub-region transferred into a histogram of eight bins, the final descriptor is of 72 dimensions. Parameter settings for SRI-DAISY are summarised in Table 3-3.

Table 3-3: Design parameters for SRI-DAISY.

Parameter	**Value**
$Ratio_{\sigma_{DAISY}}$	0.3
r/r_{local}	0.35
F_{region}	4.0

3.4.3 SRI-DAISY Implementation

The local region of each keypoint is segmented into several circular sub-regions by taking advantage of the DAISY-like polar sampled spatial arrangement. However, the descriptor is not generated in a way proposed in [60] for dense wide-baseline matching, which is not suitable for general matching tasks as a result of the drawbacks mentioned in section 3.3. In general, the rotation invariance of SRI-DAISY is achieved by arranging the spatial layout of the local region relative to θ_{po}, and the scale invariance is achieved by computing the descriptor from the scale normalised local region. This section presents how the SRI-DAISY descriptor is derived effectively without the necessity of rotating all the pixels within the local region.

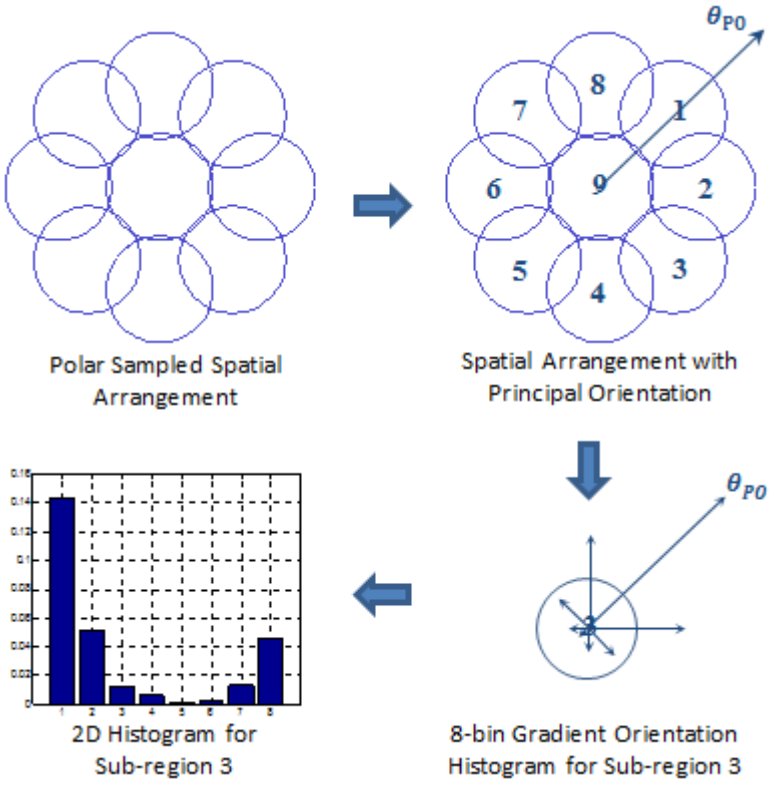


Figure 3-18: 2D histogram generation for sub-region 3.

Figure 3-18 illustrates the descriptor generation process with the polar sampled spatial arrangement of the local region. Firstly, the principal orientation θ_{po} needs to be identified, which corresponds to the orientation of the largest bin in the 2D

histogram obtained by weighting and accumulating all pixels within the local region. Secondly, the local region is segmented into nine circular sub-regions. Thirdly, with θ_{po} identified, the nine sub-regions are numbered from 1 to 9, starting from the one pointed by θ_{po} and going in a clockwise fashion, ending up with the one in the centre. Fourthly, each sub-region is transferred to an 8-bin gradient-orientation histogram. As shown in Figure 3-18, the histogram is re-ordered relative to θ_{po} with the bin in the direction of θ_{po} being in the first place of the 2D histogram. Finally, the descriptor is formed by linking together histograms of nine sub-regions in the numbered sequence. Since there are nine sub-regions with each described by an 8-bin histogram, the final descriptor is of 72 dimensions, as shown in Figure 3-19.

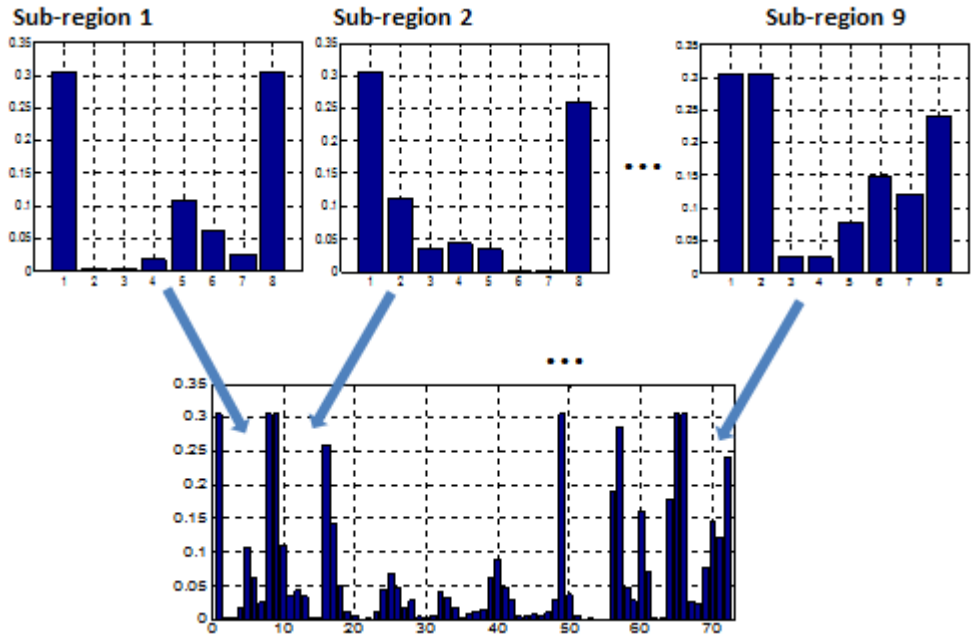


Figure 3-19: Linking together the histograms of nine sub-regions to generate a descriptor of 72-dimensions.

It is notable that, in the standard SIFT algorithm, coordinates of each pixel within the local region of a feature point should be rotated by θ_{po} to achieve rotation invariance. In this design, by taking advantage of the isotropy characteristic of the polar sampled spatial arrangement, redundantly rotating the coordinates of all pixels can be avoided

by simply arranging the eight surrounding sub-regions and the 2D histogram of each sub-region relative to θ_{po}. The arrangement can be easily achieved by figuring out the centre coordinates of eight surrounding sub-regions, with the centre pixel of the first sub-region in the direction of θ_{po}. And the rotation invariance within each sub-region is achieved by rearranging the 2D histogram of each sub-region in such a way that the bin in the direction of θ_{po} is in the first place, followed by other bins in a clockwise fashion. With the new arrangement for descriptors, the rotation of all pixels within the local region can be avoided. Furthermore, the hardware expensive sin and cos operations are saved and a descriptor can be obtained with less computational complexity.

3.4.4 Performance Comparison

The SRI-DAISY is compared with standard SIFT descriptor, in terms of both matching performance and hardware efficiency.

a. Matching Performance

The performance of SIFT and SRI-DAISY is compared, in terms of both geometric and photometric transformations, such as image rotation, scaling, viewpoint angle, image blur and illumination. The recall versus 1-precision curve is used to evaluate the descriptor performance. A perfect descriptor would give a recall equal to one for any precision. In practice, recall increases for an increasing distance threshold because the noise introduced by image transformations increases the distance between similar descriptors. A factor that leads to non-increasing recall as the threshold is increased is the distinctiveness of descriptors. In cases where images to be matched are composed of structures of high similarity, non-distinctive descriptors are unable to distinguish them thus resulting in false matches. The reason why the recall does not achieve 1.0 is because not all keypoints in the reference image are detected from the transformed images, which holds the same for the following experiments. The input to both descriptors is a square image patch that contains identical spatial information so as to eliminate the effect of different region size.

Image Rotation

To compare the performance for image rotation, a set of images with rotation angle in the range -180 and 170 degrees is used, covering 360 degrees. The image rotation is obtained by rotating the camera around its optical axis. Example images are shown in Figure 3-20.

Figure 3-20: Example images in the dataset used for evaluation of image rotation.

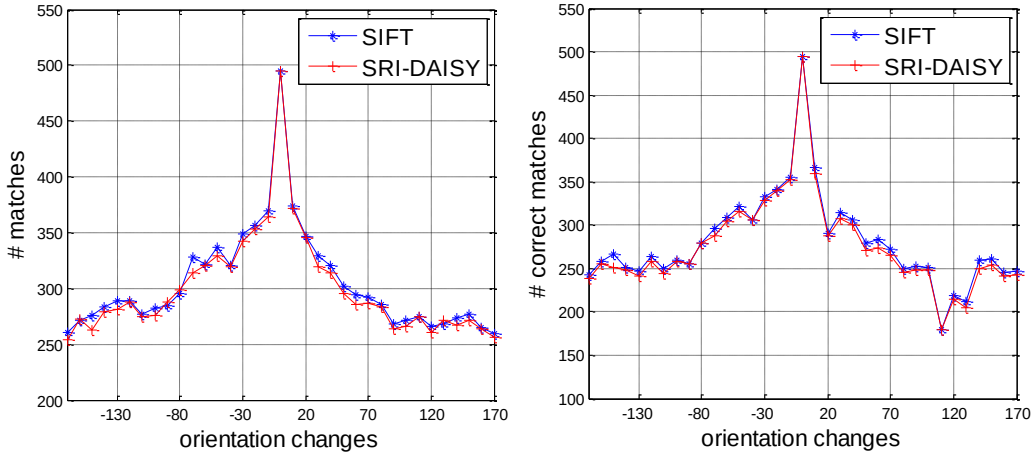


Figure 3-21: The number of both total and correct matches for a set of images with orientation in range -180^{o} to 170^{o}.

Figure 3-21 shows that the number of matches of SRI-DAISY is slightly below that of SIFT. The recall versus 1-precision curves for rotation in range 10^{o} to 60^{o} are displayed in Figure 3-22, which shows that both curves are horizontal at a similar recall value of around 0.9, indicating that both descriptors have a similar robustness to image rotation.

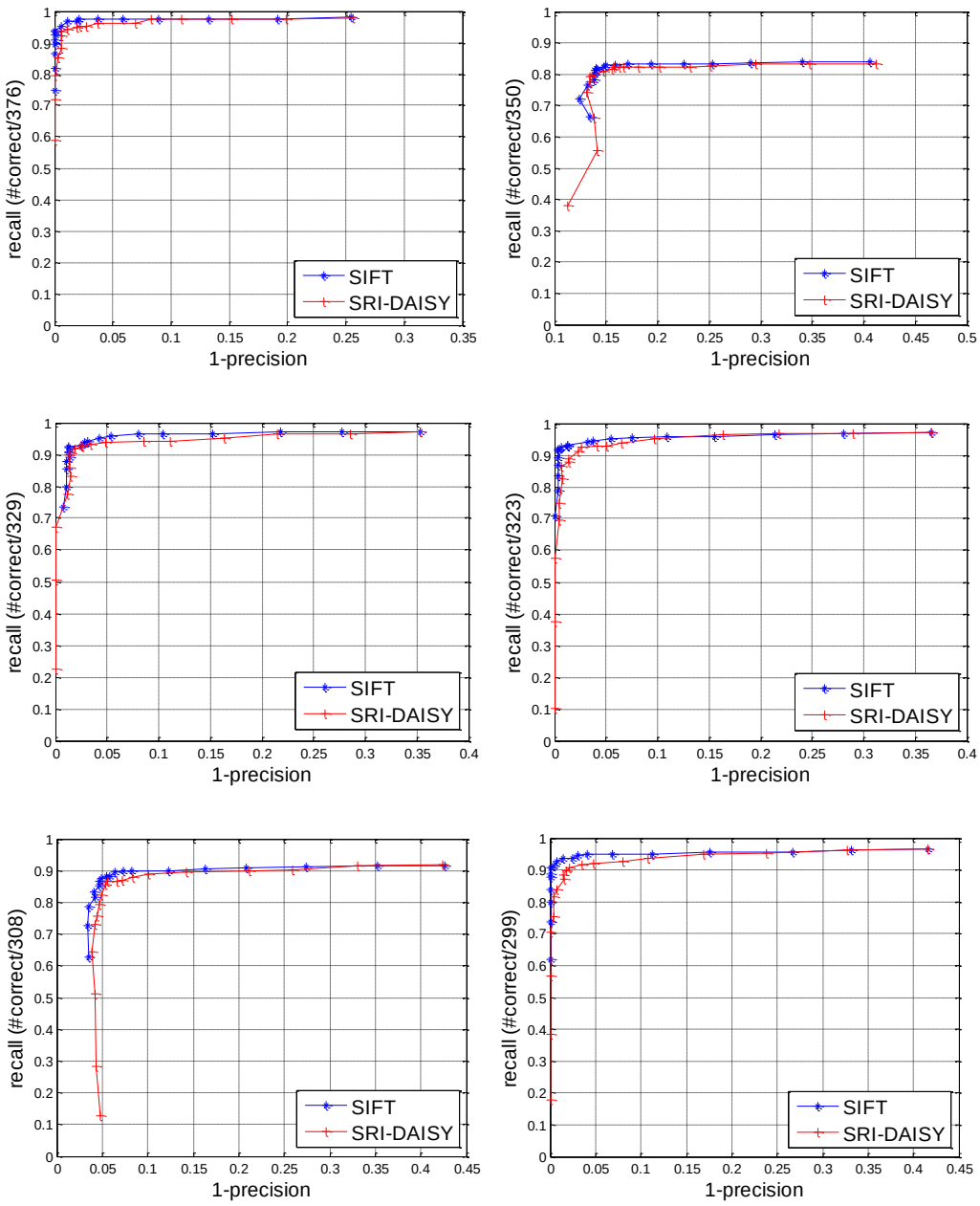


Figure 3-22: The recall versus 1-precision curve for image rotation of 10^{o} to 60^{o}.

Scale Invariance

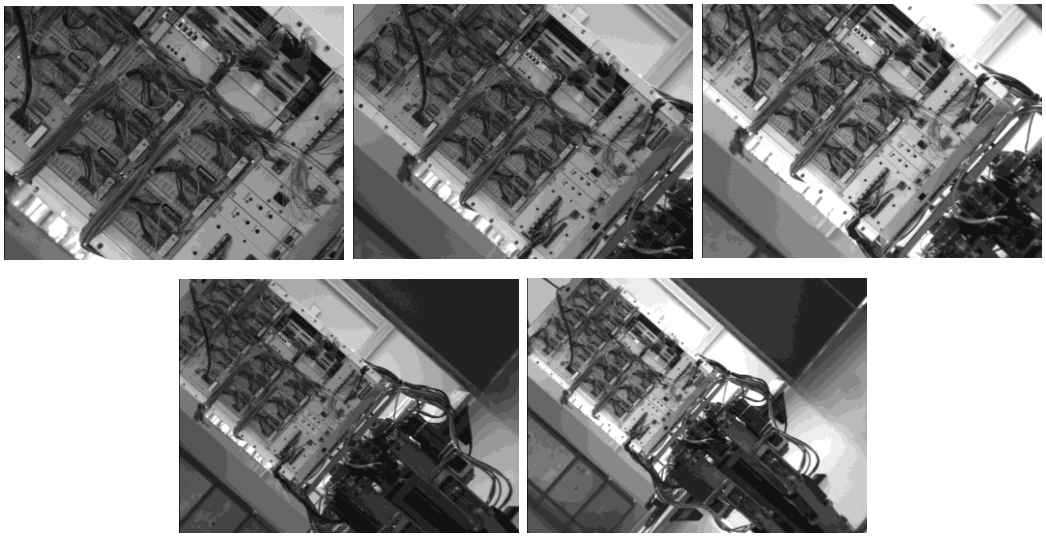


Figure 3-23: Dataset used for evaluation of scale changes.

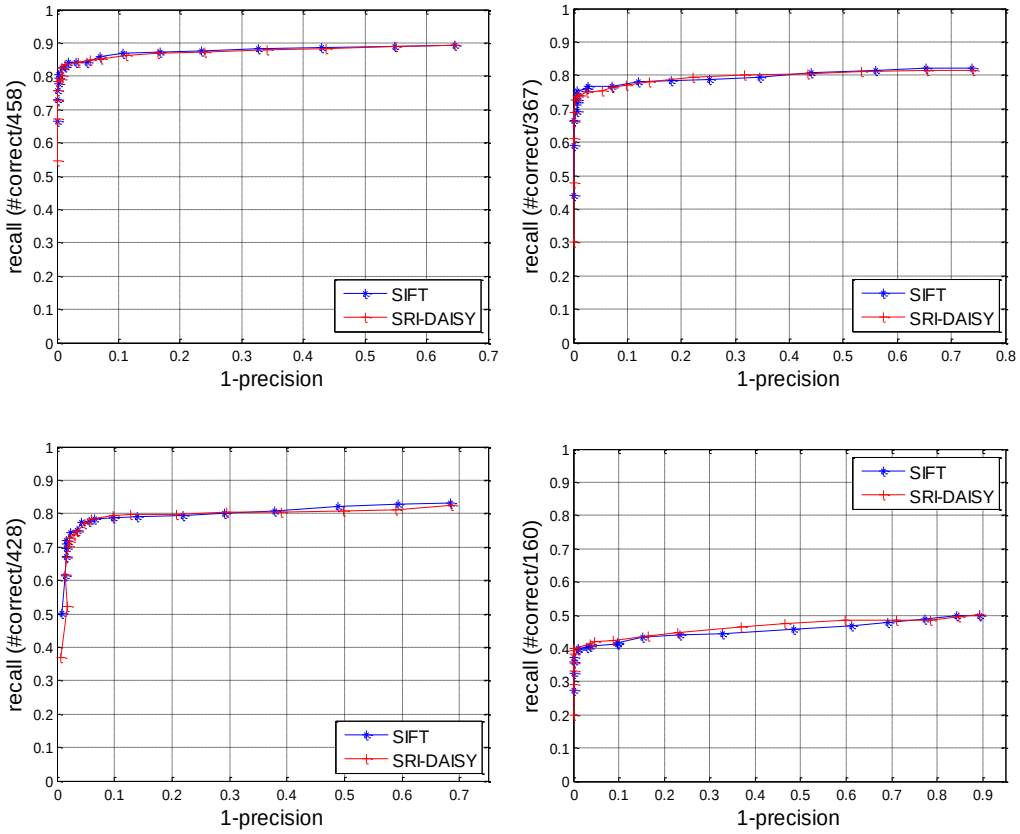


Figure 3-24: Performance comparison between SIFT and SRI-DAISY on a set of images with scale changes of a factor in range 1.47 to 3.75.

Scale change is acquired by changing the camera zoom. This section compares the descriptors for scale changes in range 1.47 to 3.75. As shown in Figure 3-23, the leftmost image is the reference that is matched against the other four images with different scaling factors. Both the SRI-DAISY and the SIFT descriptor are generated using the image patch of the same size and perform virtually the same, as shown in Figure 3-24.

Viewpoint Change

Viewpoint change is acquired by rotating the camera around the axis that is perpendicular to its optical axis. Neither SIFT nor SRI-DAISY is fully invariant to viewpoint changes. The partial invariance to such type of transformation is achieved by the overall robustness of the descriptor. It can be seen from Figure 3-26 that the curves of both descriptors are horizontal at a similar recall value of around 0.8 when the viewpoint angle is below 40 degrees, but degrades significantly afterwards. Lowe has pointed out that invariance to viewpoint changes of greater than 40 degrees is unnecessary [10], because training views are best taken at least every 30 degrees in order to capture non-planar changes and occlusion effects for 3D objects. Therefore, both SIFT and SRI-DAISY are robust enough for matching images with a viewpoint angle of no greater than 30 degrees.

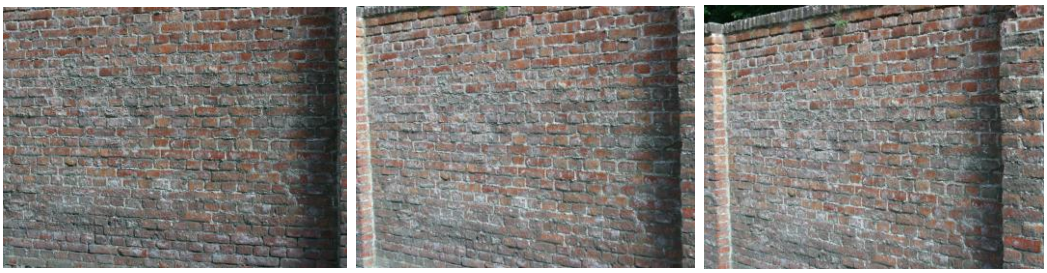


Figure 3-25: Dataset used for evaluation of viewpoint change.

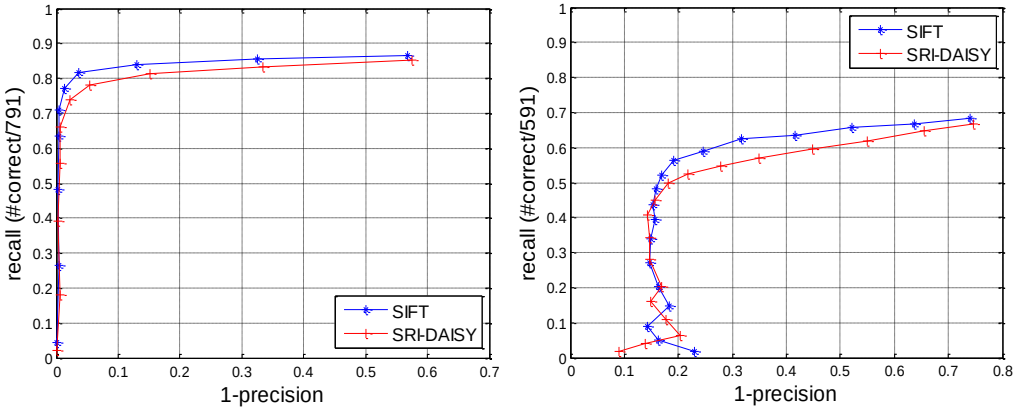


Figure 3-26: Performance comparison between SIFT and SRI-DAISY on a set of images with viewpoint changes of 30 and 40 degrees.

Image Blur

Image blur is introduced by changing the camera focus, which causes the image intensities and local structures change in an unpredictable way [27].

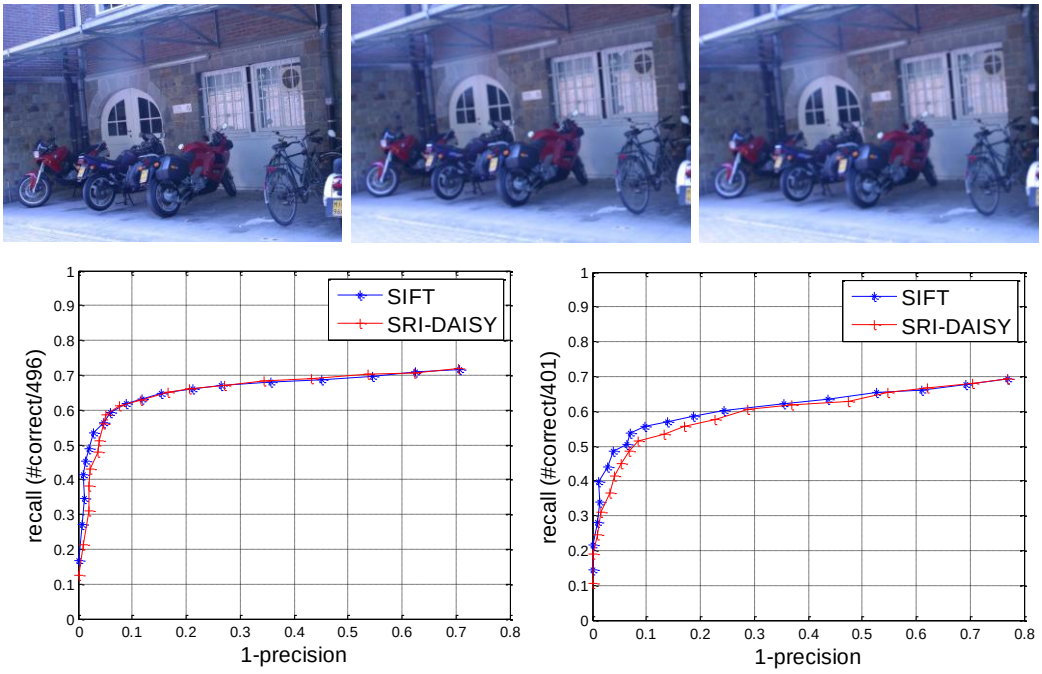


Figure 3-27: Performance comparison between SIFT and SRI-DAISY of image blur on a set of structured images.

It can be seen from Figure 3-27 and Figure 3-28 that both descriptors are partially robust to image blur, even for the more challenging textured scene where blur makes regions nearly identical.

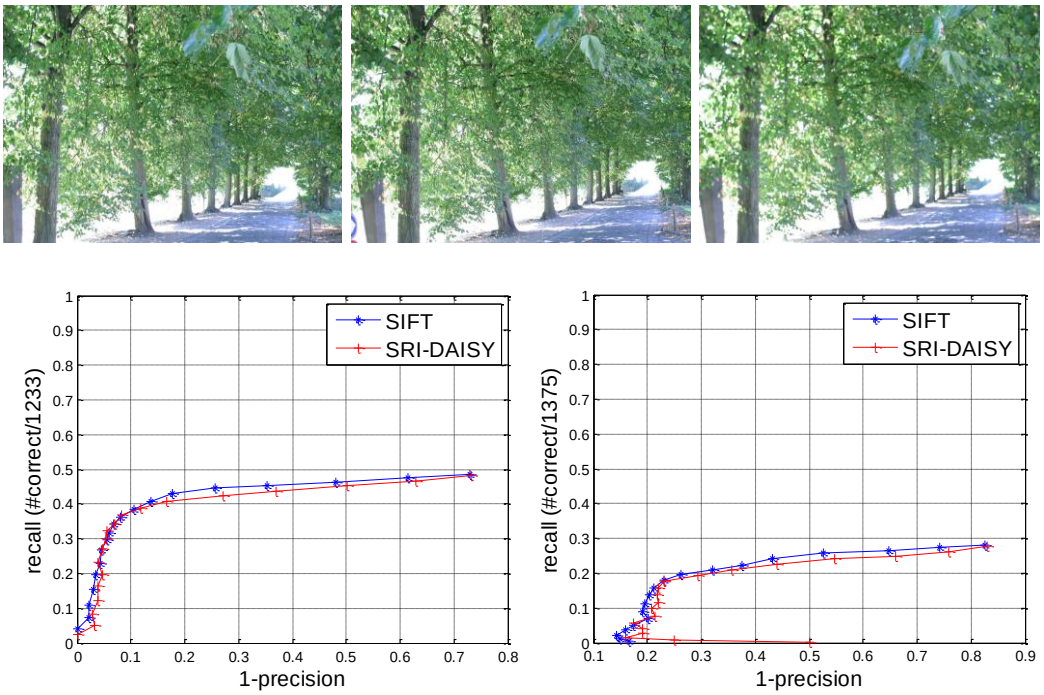


Figure 3-28: Performance comparison between SIFT and SRI-DAISY of image blur on a set of textured images with repeat textures.

Illumination

Figure 3-29 shows the matching results for illumination changes obtained by varying the camera aperture. Both descriptors are normalised to reduce the effects of illumination changes, and the curve of SRI-DAISY is slightly below that of SIFT.

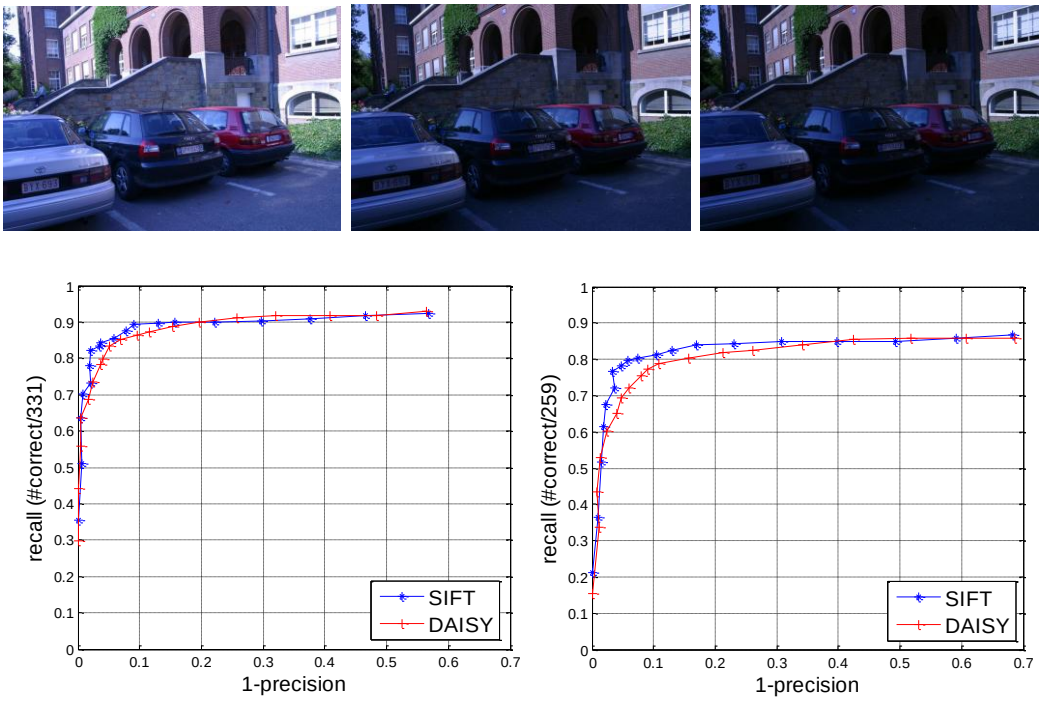


Figure 3-29: Performance comparison between SIFT and SRI-DAISY on a set of images with illumination changes.

It can be seen from the above mentioned experiments that the SRI-DAISY descriptor has achieved a dimension reduction while providing comparable performance to the standard SIFT descriptor.

b. Hardware Efficiency

Figure 3-30 shows the overview of descriptor generation process for standard SIFT, standard DAISY, O-DAISY and SRI-DAISY.

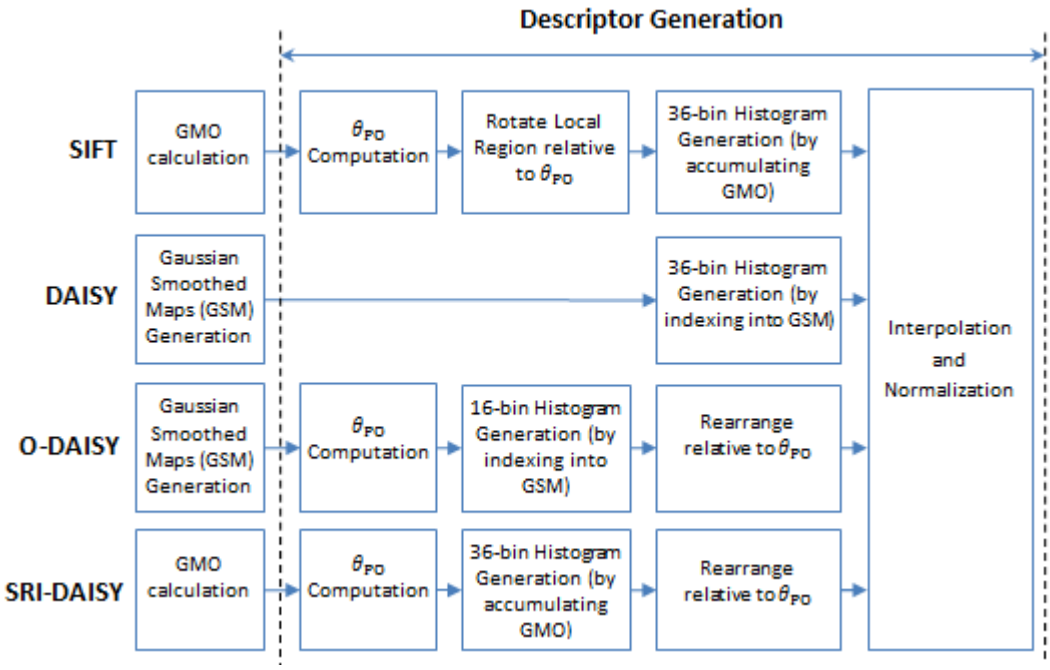


Figure 3-30: Overview of descriptor generation process for standard SIFT, standard DAISY, O-DAISY and SRI-DAISY.

To achieve rotation invariance, the principal orientation (θ_{po}) is first assigned to each feature. In the standard SIFT, all pixels within the local region centred on the feature point has to be rotated relative to the principal orientation (θ_{po}) following Equation (3.5) so as to achieve rotation invariance, as shown in Figure 3-31. This process includes complex sin and cos operations that are expensive to be implemented on FPGA devices. Therefore, the standard SIFT descriptor is inefficient for FPGA implementation due to its rotation scheme.

$$x' = x \cdot cos(\theta_{po}) + y \cdot sin(\theta_{po})$$
$$y' = y \cdot cos(\theta_{po}) - x \cdot sin(\theta_{po}) \quad (3.5)$$

where (x, y) and (x', y') are the pixel coordinates before and after rotation, respectively.

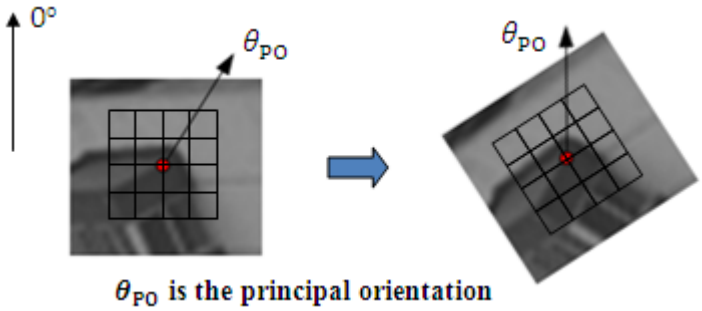


Figure 3-31: Rotation of local region relative to the principal orientation (θ_{po}) for rotation invariance of standard SIFT, assuming upright direction is 0^{o}.

The SRI-DAISY tackled the drawbacks of the rotation scheme for standard SIFT by taking advantage of the spatial arrangement of the descriptor, as has been presented in section 3.4.3. In SRI-DAISY, sub-regions are first summarised into 36-bin histograms that are further re-ordered relative to θ_{po} for rotation invariance. Compared with standard SIFT, the computational complexity of descriptor generation is reduced. Besides, in the standard SIFT algorithm, boundary has to be defined for each square sub-region to process the pixels within it for histogram generation. However, the necessity of identifying the boundary of each sub-region for histogram generation can be avoided by applying a Gaussian function to each circular sub-region with coefficients outside the boundary set to zero. As a result, the computational complexity is further reduced.

In both standard DAISY and O-DAISY, Gaussian smoothed orientation maps of all discrete directions have to be buffered for fast indexing in descriptor generation process. For SRI-DAISY, it only needs to buffer the GMOs of the scales from which the features are detected. Compared with standard DAISY and O-DAISY, SRI-DAISY has achieved a significant memory reduction.

3.5 A Novel Matching Strategy

This section proposes a novel feature matching strategy that is not only more accurate in matching, but also more efficient to be implemented on FPGA devices.

3.5.1 Existing Matching Strategies

The matching process is one of the fundamental tasks in computer vision and takes place among the keypoints associated with descriptors. A good set of correspondences between images is essential in order to carry out further tasks. In general, there are three widely used matching strategies: 1) Threshold based matching 2) Nearest neighbour based matching 3) Distance ratio based matching. The matching strategy using a global threshold does not perform well due to the fact that the distinctiveness of keypoint varies. The matching strategy based on the nearest neighbour performs better than the threshold based matching, but it finds every keypoint in the input image a matched keypoint from the reference image, which leads to many incorrect matches. To deal with the drawbacks of the previous two matching strategies, Lowe [10] proposed a new matching strategy under the assumption that a correct match need to have the closest neighbour significantly closer than the closest incorrect match. Therefore, a match is accepted if the ratio between the closest neighbour and the second closest neighbour is smaller than the pre-defined threshold, as shown in Equation (3.6).

$$\frac{\sqrt{\sum_{i=0}^{Dim}[d_a(i) - d_b(i)]^2}}{\sqrt{\sum_{i=0}^{Dim}[d_a(i) - d_c(i)]^2}} < 0.8 \tag{3.6}$$

where d_a is a descriptor from the input image, d_b and d_c is the closest and the second closest neighbour from the reference image, respectively. Dim denotes the descriptor dimension, and i is the index to each dimension of the descriptor.

3.5.2 A Novel Matching Strategy

Inspired by the three existing matching methods, a novel matching strategy that is well balanced between performance and computation efficiency is proposed. Experiments are conducted to show the performance comparison between the proposed method and Lowe's distance ratio based matching.

Instead of computing the Euclidean distance between descriptors, the novel matching method focuses on the difference (Δd) between each dimension of the pair of descriptors under consideration, as shown in Equation (3.7). Δd represents the similarity between dimensions, and lower value indicates higher similarity.

$$\Delta d(i) = d_{ref}(i) - d_{inp}(i) \tag{3.7}$$

where i is the index to the dimension of descriptors and is in range [1, 72].

A pair of descriptors is accepted as correct matches only to meet the following two conditions:

1. The potential pair of match need to have the ratio of the largest $N_{\Delta d}$ to the dimensionality (Dim_{72}) of the descriptor greater than a pre-defined threshold, which can be expressed as $\frac{N_{\Delta d}}{Dim_{72}} > Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$. $N_{\Delta d}$ is the number of dimensions with Δd below a pre-defined threshold $Thr_{\Delta d}$.
2. The ratio of $N_{\Delta d_{second}}$ to $N_{\Delta d_{closest}}$ needs to below a pre-defined threshold, which can expressed as $\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}} < Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$, where $N_{\Delta d_{second}}$ and $N_{\Delta d_{closest}}$ is $N_{\Delta d}$ of the second-closest match and that of the closest match, respectively.

The first condition is a combination of the threshold based and the nearest neighbour based matching, which ensures that the potential pair of matches is of high similarity and selects only the best match with the largest $N_{\Delta d}$ above a pre-defined threshold. The second condition has the same basic idea with that of the distance ratio based matching and rejects the matches of similar distances. The closest match is defined as the pair of descriptors with the largest $N_{\Delta d}$. The number of incorrect matches will

be reduced as a result of the second condition, and hence the precision is improved. The ratio is taken as the closest to the second-closest for the SIFT-based matching, but it is the second-closest to the closest for this strategy.

Experiments are conducted to determine the following parameters: $Thr_{\Delta d}$, $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ and $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$.

a. Parameters for Similarity Measurement of Descriptors

The similarity threshold ($Thr_{\Delta d}$) is first estimated experimentally using a database of over 1,000 correctly matched descriptors from a diverse range of scenes with different transformations. Figure 3-32 shows the probability density function (PDF), in terms of Δd between each dimension, which shows that Δd of about 90% dimensions are under 0.05, assuming that the descriptor has been normalised and each dimension is in range [0,1].

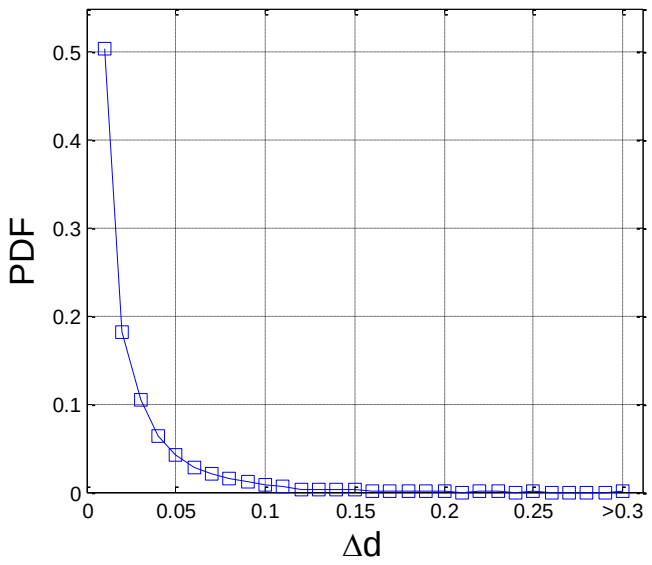


Figure 3-32: The probability density function of the distance between each dimension of descriptors. The data is obtained using a database of over 1,000 pairs of descriptors that are correctly matched.

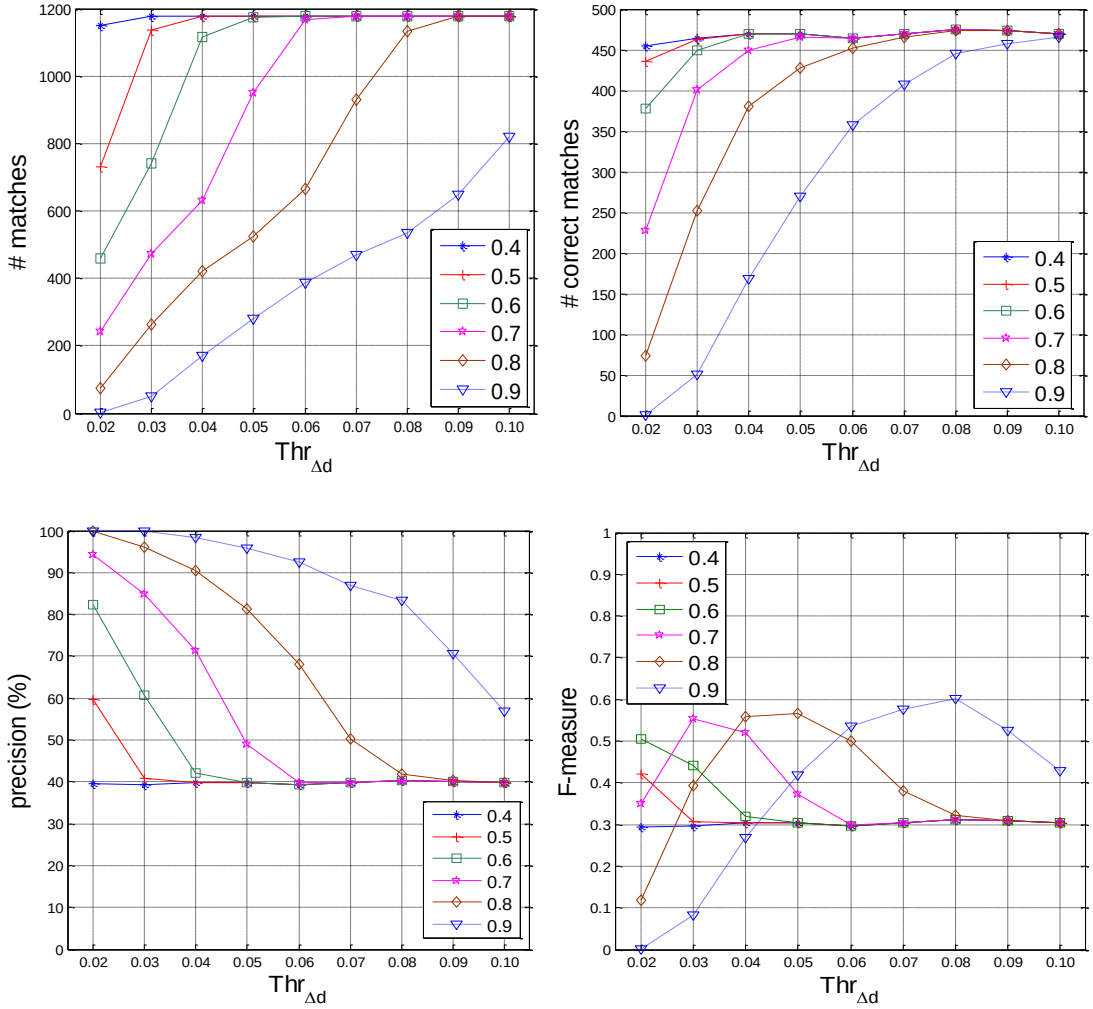


Figure 3-33: Matching performance as a function of distance threshold $Thr_{\Delta d}$ for different $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ in range [0.4, 0.9] of interval 0.1.

Experiments are first conducted to check the relationship between $Thr_{\Delta d}$ and $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ and their impact on the matching performance. Figure 3-33 shows the matching performance as a function of $Thr_{\Delta d}$ for different $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ in range [0.4, 0.9] of interval 0.1. The ratio of $N_{\Delta d}$ of the second-closest match to that of the closest match is not considered in this experiment. In general, for a given $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$, the number of both matches and correct matches increases as the distance threshold $Thr_{\Delta d}$ is relaxed, but the precision decreases as a result of the number of incorrect matches that increases faster than correct matches. Different $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ achieves the

best F-measure with different $Thr_{\Delta d}$, which is due to the difference in PDF of correct and incorrect matches as illustrated by the two examples given in Figure 3-34.

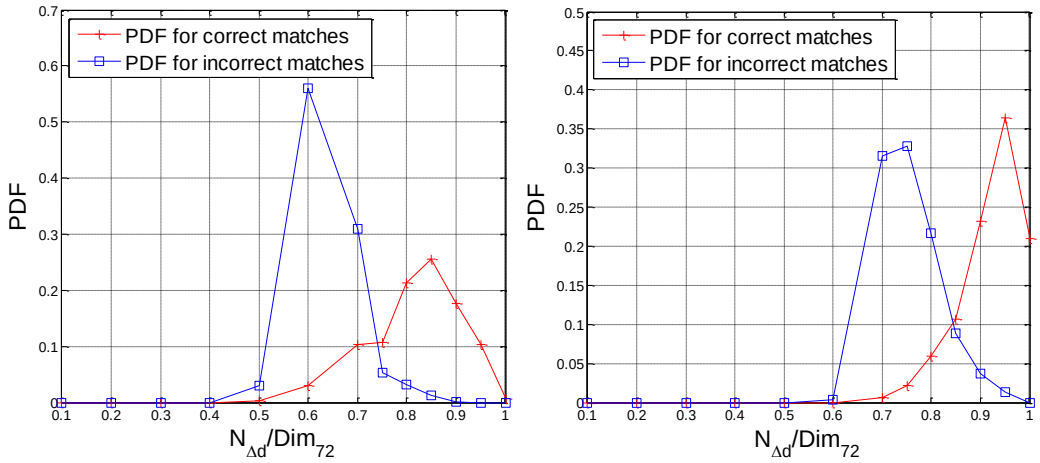


Figure 3-34: The left image shows the PDFs for $N_{\Delta d} = 0.03$. The right image shows the PDFs for $N_{\Delta d} = 0.05$.

Figure 3-34 shows the PDF for correct and incorrect matches, in terms of the ratio of $N_{\Delta d}$ to Dim_{72}. The blue line with square marker shows the PDF of this ratio for incorrect matches, and the red line with plus marker is for correct matches. In general, the correct matches have a PDF centred at a higher ratio than the incorrect matches, and the centres for both correct and incorrect matches vary with $Thr_{\Delta d}$. For $Thr_{\Delta d}$=0.03, if $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ is larger than 0.7, the number of correct matches decreases faster than the incorrect matches, resulting a rise in precision but a drop in recall. If $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ is below 0.7, the number of incorrect matches increases faster than correct matches, and hence a higher recall but a lower precision. The best F-measure which is a balance between recall and precision is achieved by $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$=0.7. Therefore, all the matches with the ratio of $N_{\Delta d}$ to Dim_{72} below $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$=0.7 are rejected, which eliminates 91% of the incorrect matches while discarding about 13% of correct matches. For $Thr_{\Delta d}$=0.05, all the matches with the ratio of $N_{\Delta d}$ to Dim_{72} below $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$=0.8 are rejected, which eliminates 87% of the incorrect matches while

discarding 14% of correct matches. In general, $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ needs to be relaxed as $Thr_{\Delta d}$ is relaxed, and the idea is not restricted to the above mentioned two examples.

b. Ratio Threshold for Incorrect Matches Rejection

This section introduces the impact of $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$ on the matching performance. All the matches with the ratio between the second-closest neighbour and the closest neighbour greater than $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$ are rejected. Without the threshold on the ratio of $N_{\Delta d_{second}}$ to $N_{\Delta d_{closest}}$, keypoints that do not have corresponding matching point are also assigned a matching point, which leads to many incorrect matches.

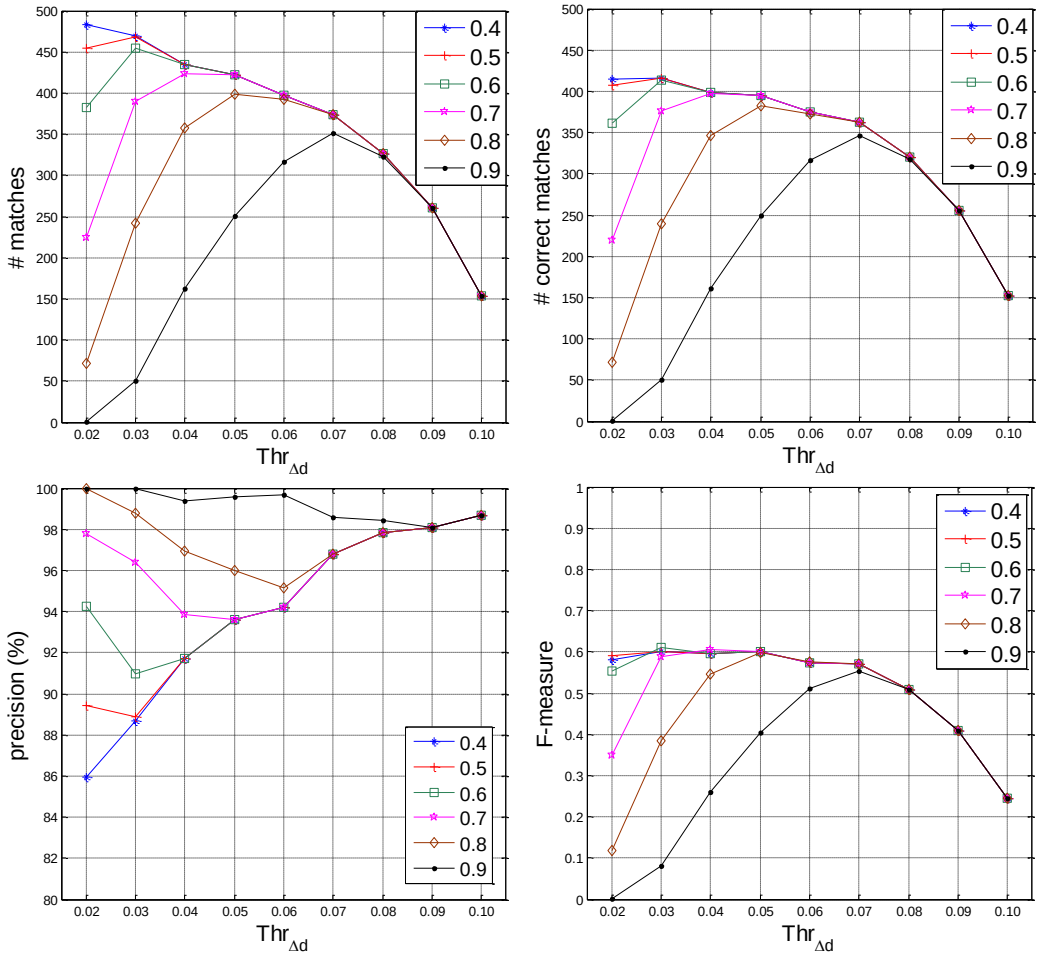


Figure 3-35: Matching performance as a function of distance threshold $Thr_{\Delta d}$ for different $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ in range [0.4, 0.9] of interval 0.1.

Figure 3-35 shows the matching results of $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$ = 0.9. The number of total matches is reduced significantly when compared with the results from Figure 3-33, resulting in a significant improvement on precision. The number of both total and correct matches increase at the beginning as the distance threshold $Thr_{\Delta d}$ is relaxed, but drops beyond a certain point, which can be explained by the PDFs of correct and incorrect matches as a function of $\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}$, as shown in Figure 3-36.

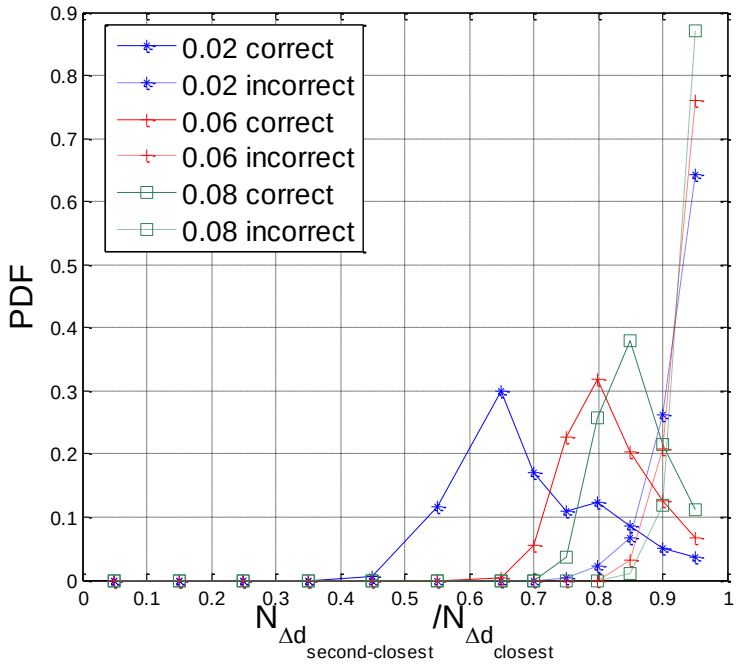
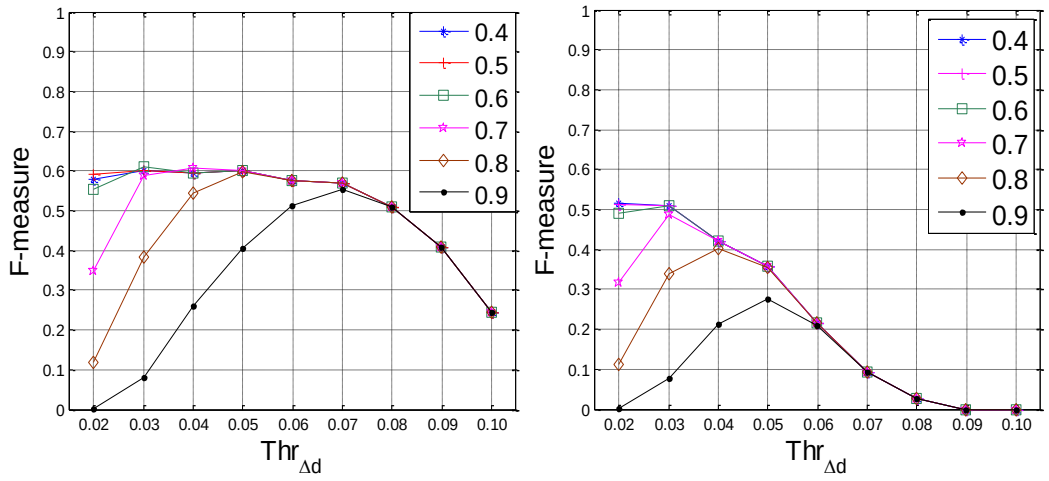


Figure 3-36: The probability that a match is correct can be determined by taking the ratio of $N_{\Delta d_{second}}$ to $N_{\Delta d_{closest}}$.

Figure 3-36 shows the PDFs for correct and incorrect matches as a function of the ratio of $N_{\Delta d_{second}}$ to $N_{\Delta d_{closest}}$ for three different $Thr_{\Delta d}$. The solid lines show the PDF for correct matches of different $Thr_{\Delta d}$, whereas the dashed lines are for incorrect matches. In general, correct matches have a PDF centred at a lower ratio than that for incorrect matches. The majority of incorrect matches has the ratio of $N_{\Delta d_{second}}$ to $N_{\Delta d_{closest}}$ larger than 0.9. Therefore, by discarding matches of $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$>0.9,

about 90% of incorrect matches are eliminated while discarding a certain number of correct matches, resulting in a slight decrease in the number of correct matches. Besides, correct matches for smaller $Thr_{\Delta d}$ have a PDF centred at a lower ratio, and hence are on the average more distinctive. The centre of correct matches and incorrect matches for larger $Thr_{\Delta d}$ are close to each other, leading to a larger number of correct matches discarded as incorrect. Therefore, by setting the ratio threshold to 0.9, the number of correct matches for larger $Thr_{\Delta d}$ drops faster. If the ratio threshold is lowered to 0.8, almost all the incorrect matches are eliminated while a large number of correct matches are discarded, leading to the precision of nearly 100% but a significant drop in recall.



(a) $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}} = 0.9$ (b) $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}} = 0.8$

Figure 3-37: Matching performance for different threshold value on the ratio of $N_{\Delta d_{second}}$ to $N_{\Delta d_{closest}}$.

Figure 3-37 shows the matching performance for $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$ = 0.9 and 0.8 as a function of the distance threshold $Thr_{\Delta d}$ for $Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$ in range [0.4, 0.9] of interval 0.1. The overall matching performance drops with the decrease of $Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$ as a result of the significant reduction in the number of correct matches.

c. Comparison with Matching Strategy from SIFT

Three sets of parameters listed in Table 3-4 are compared with the distance ratio based matching.

Table 3-4: Parameters for the novel matching strategy.

Setting	$Thr_{\Delta d}$	$Thr_{\frac{N_{\Delta d}}{Dim_{72}}}$	$Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$
1	0.03	0.6	0.9
2	0.04	0.7	
3	0.05	0.8	

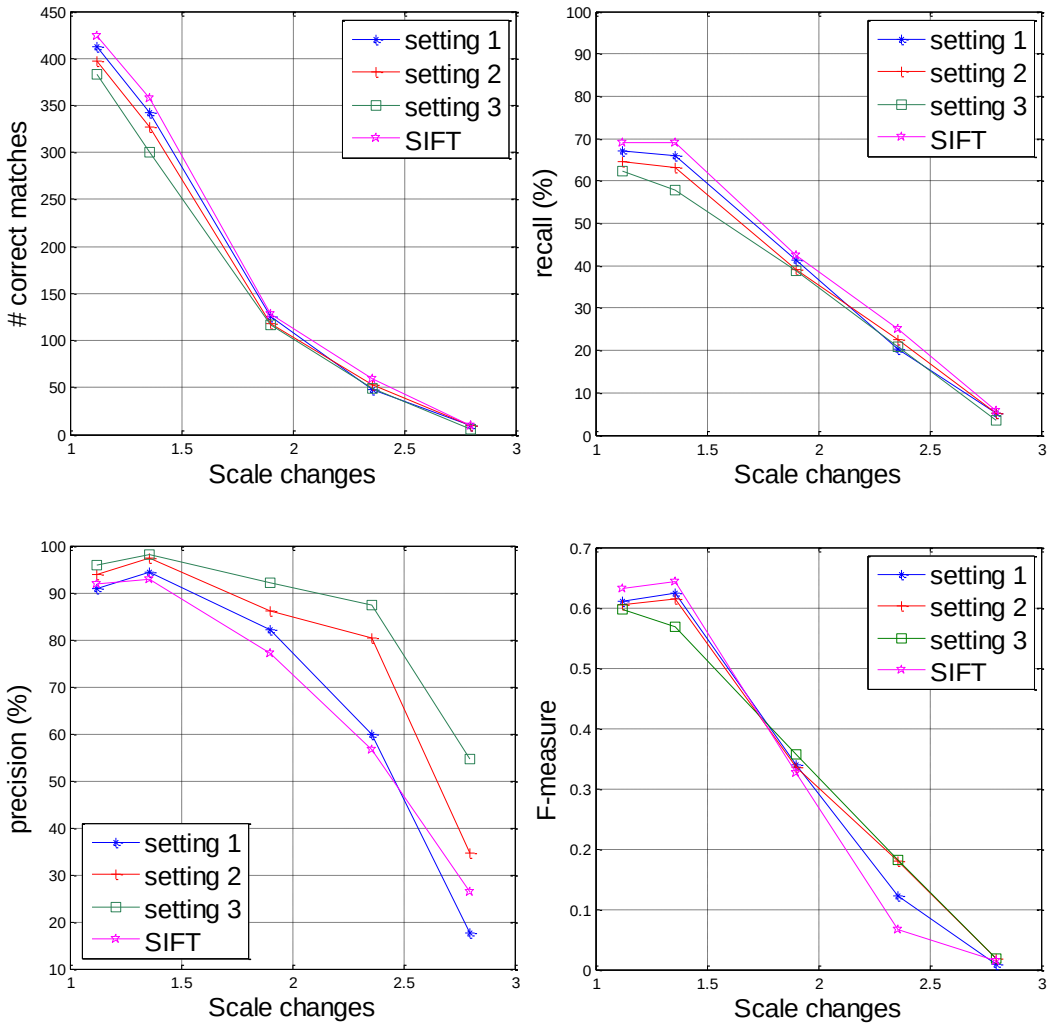
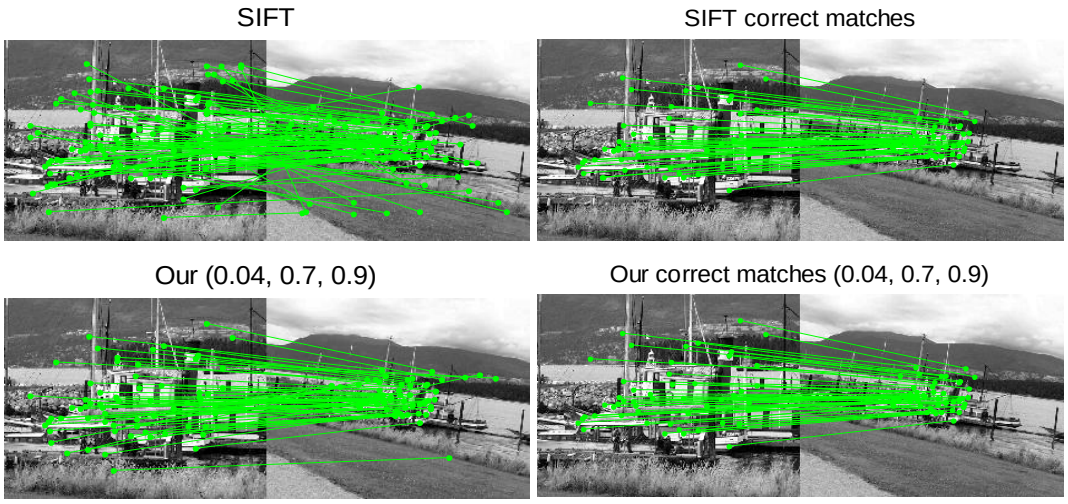
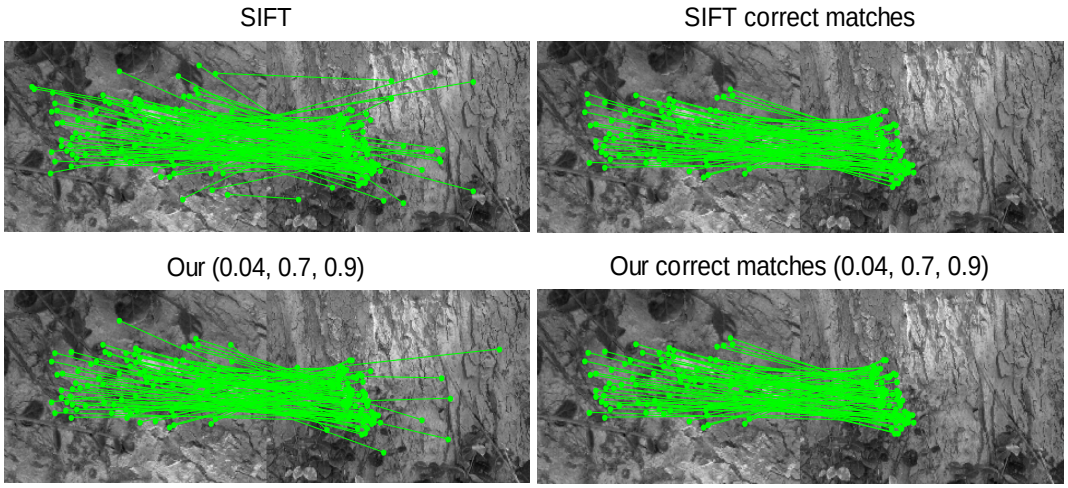


Figure 3-38: Comparison between the proposed matching strategy and the distance ratio based matching proposed by Lowe.

Figure 3-38 shows the experimental results from the boat sequence. In general, SIFT has the highest recall. However, the precision of setting 2 and setting 3 are significantly superior to that of SIFT, especially in presence of large transformations where the distance between descriptors is on the average large. It can be seen from the F-measure that setting 2 achieves the best balance between recall and precision. Setting 3 is suggested for applications that concern more about the matching precision.



(a) Matching performance comparison between the distance ratio based matching proposed in SIFT and our novel matching strategy on the boat set.



(b) Matching performance comparison between the distance ratio based matching proposed in SIFT and our novel matching strategy on the tree set.

Figure 3-39: Performance comparison (a) structured scene; (b) textured scene.

Figure 3-39 shows the matching results conducted on two pairs of images of different scene types. In Figure 3-39(a), for the distance ratio based matching proposed by Lowe, there are 104 initial matches, 59 of which are correct, giving the precision of 56.73%. For the novel method, there are 66 total matches, 53 of which are correct, giving precision of 80.30%. In Figure 3-39(b), for the distance ratio based matching, there are 101 matches, 77 of which are correct, giving the precision of 76.24%. For the novel method, 74 out of the 85 initial matches are correct, giving precision of 87.06%. Despite of the incorrect matches that exist in both methods, it is obvious that the novel method obtains higher matching precision.

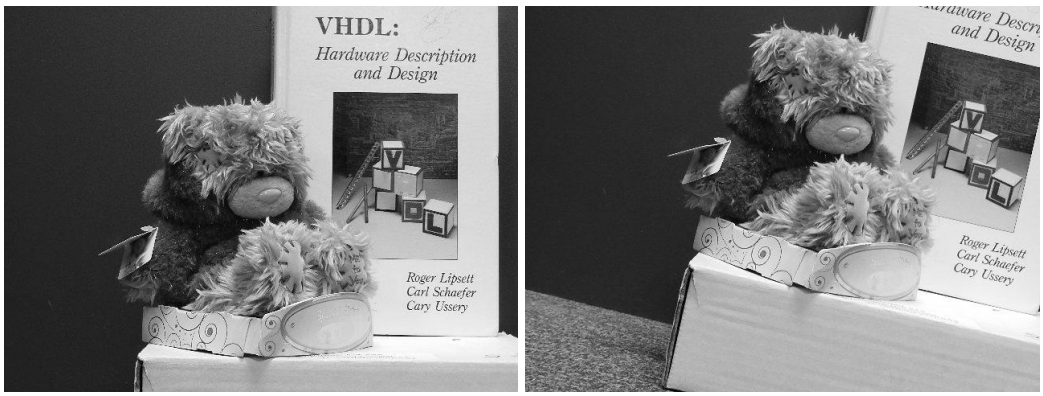
The novel matching strategy is more robust to partial occlusion than SIFT. In the presence of partial occlusion, parts of the histogram can be very different for the sub-regions occluded even for good matches. This will lead to a significant change in the Euclidean distance between descriptors. However, the matching result is less likely to be affected for the novel method. This is mainly because the novel method does not reply on the overall Euclidean distance between descriptors, but is closely related to the distance (Δd) between each dimension, which allows parts of the descriptor to be significantly changed, and hence allows the local region to be partially occluded. In short, the matching strategy proposed in this section not only achieves comparable performance with that of SIFT, but also is more robust to partial occlusion and is computationally more efficient.

d. Advantage in Application for Video Stabilisation

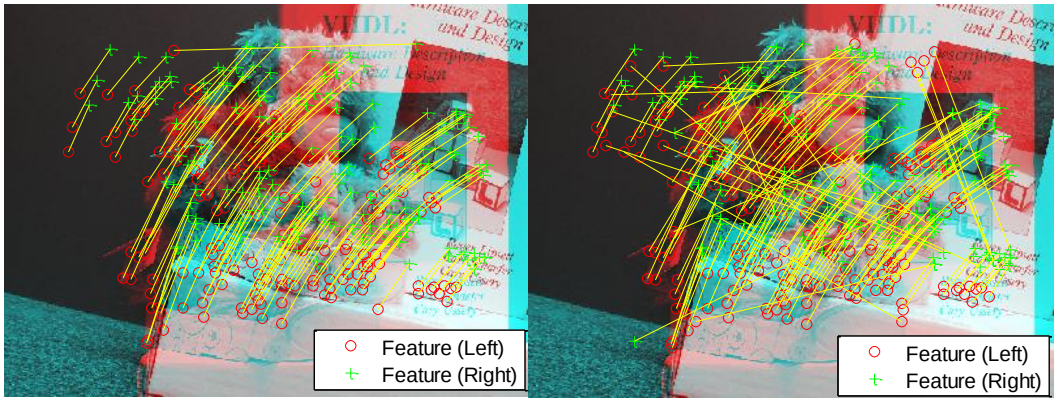
The novel matching strategy is beneficial to applications, such as video stabilisation. In video stabilisation, RANSAC (RANdom SAmple Consensus) and least square are usually utilised to estimate motion vectors. The least square is optimally fitted to all matches, including both inliers and outliers. Therefore, least square does not perform well when there is a larger portion of outliers in the total number of matches, and hence higher matching precision is desirable. RANSAC, on the other hand, is computed from inliers, and the processing time of RANSAC is proportional to the number of iterations for model parameters estimation. If the number of iterations is limited, the obtained parameters may not be optimal, and it may not even be the one that fits the input data. Higher matching precision provides a larger probability of

choosing inliers, and hence requires less iteration to produce a motion model. Therefore, higher matching precision is desirable for model estimation with higher precision and less processing time.

Figure 3-40(a) shows a pair of images that are matched by using the novel matching strategy and the distance ratio based matching, respectively. The original matching results are given in Figure 3-40(b).



(a) Image pair under consideration



(b) Original matching results. Left: the novel strategy; Right: distance ratio based method.

Figure 3-40: Matching performance comparison between the novel strategy and the distance ratio based method,

Figure 3-41 shows the comparison results with least square employed for motion estimation. The left images shown in Figure 3-40(a) is warped using the transformation matrix estimated with least square. The corresponding Mean Square Error (MSE) is 28.3125 and 48.1472, respectively. The MSE quantifies the difference between an obtained result and its expected value. It measures the average of the square of the error, where the error is the amount by which the result differs from the expected value. It is obvious that high precision matching is beneficial to least square based modelling fitting

Figure 3-41: The left column shows the warped image of the novel matching strategy and the right column is for the distance ratio based matching.

Figure 3-42 shows the number of inliers identified by RANSAC as a function of the number of iterations. The first three boxplots are for the novel matching strategy, and the last three boxplots are for the distance ratio based matching from SIFT. The

number of iterations is set to 5, 20, and 50, respectively. It can be seen from Figure 3-42 that the novel method is more stable than nearest based matching.

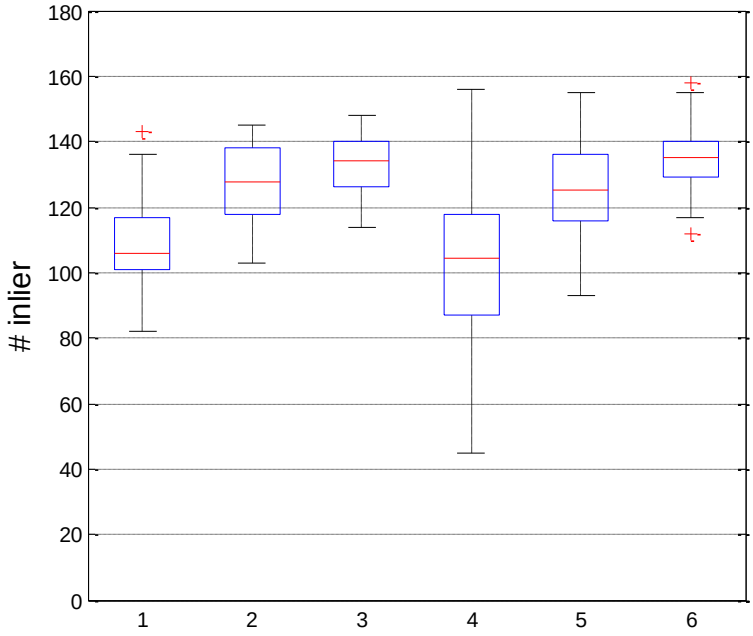


Figure 3-42: The number of inliers as a function of the number of iterations for RANSAC.

Figure 3-43 shows the red-cyan colour composite of the overlay of the original right image and the transformed left image. RANSAC is conducted on both the novel matching strategy and the distance ratio based matching, and the results are shown on the left column and right column, respectively. Each row corresponds to one of the three number of iterations evaluated. It can be seen from Figure 3-43 that two images are better aligned using the novel method. A larger number of iterations are required when applying RANSAC to the matches from the distance ratio based method for higher parameter accuracy. The novel matching strategy requires less iterations for model estimation, and hence a reduction in processing time.

Figure 3-43: Comparison of transformation accuracy by showing the red-cyan colour composite of the overlay of the original right image and the transformed left image.

3.6 Summary

This chapter proposed to replace the grid layout of SIFT with the log-polar spatial arrangement of DAISY. The SRI-DAISY is improved upon the standard DAISY that is initially proposed for dense wide-baseline matching, and is invariant to both rotation and scale changes. Compared with O-DAISY, the orientation precision is

improved from 22.5^o to 10^o for each discrete direction. By employing the log-polar spatial arrangement, shifting all pixels within the local region has been replaced by simply arranging both the spatial layout and histogram of each sub-region relative to θ_{po}, with which the complex sin and cos functions are avoided. Besides, the necessity of identifying the boundary of each sub-region for histogram computation is also avoided. By arranging the local region into nine sub-regions, the descriptor dimension is reduced from 128 to 72, which reduces the memory requirement to buffer descriptors. The SRI-DAISY achieves comparable performance with standard SIFT and is more efficient to be implemented using hardware, in terms of both computational complexity and memory usage.

Besides, a novel keypoint matching strategy has been proposed in this chapter, which provides higher precision than the distance ratio based matching. By using the novel matching strategy, both the squaring operations and the square root computation are avoided, and hence the novel matching strategy is more efficient to be implemented on hardware devices.

Chapter 4 Design Considerations

4.1 Introduction

In this chapter, design parameters are considered to configure the system with high performance and low hardware resource usage. Besides, detailed error analysis is performed to see the effect of fixed-point arithmetic on the design performance to enable efficient and accurate hardware architecture. Simulation results are presented to compare the performance of the proposed processing core with the software model with floating-point accuracy.

4.2 System Configuration

Prior to defining the hardware architecture for the optimised SIFT algorithm, a series of experiments have been done in order to find the best set of parameters for the SIFT based matching system. Each experiment aims to evaluate a particular aspect of the method. The system throughput is a most relevant performance measure, and a set of possible configurations is evaluated to establish a parameter combination that retains a good performance while it achieves as close as possible to real-time.

In this section, all the results are obtained by matching a wide range of images against themselves, but with various combinations of rotation and translation movements. Therefore, the mapping relationship between a pair of images is known or can be computed. The homography between the reference image and other images in the same set of data are known and accurate, and can be used to provide ground truth matches for the detector.

4.2.1 Evaluation Criterion

Real applications need distinctive and repetitive keypoints that can be differentiated from the others and can be repeatedly detected in different views of the same scene or object. Repeatability is one of the most important performance evaluation criteria for the stability of feature detectors. It measures the ability of a detector to extract the same feature points across images irrespective of imaging conditions. For a given pair of images, the repeatability rate is computed as the ratio of the number of

correspondences to the smaller number of detected interest points in the commonly visible region of the pair of images, as shown below.

$$Repeatability = \frac{\#\ correspondences}{min(N1, N2\ )} \tag{4.1}$$

where $N1$ and $N2$ are the number of keypoints detected from the commonly visible part of the pair of images, respectively.

Finding correspondences between image pairs using interest points are based on the assumption that salient interest points will be repeatedly detected in both images. The corresponding interest points are expected to be precisely localised on the same scene element, and the associated surrounding region is supposed to cover the same part of the scene. Therefore, the corresponding interest points are regarded as potential features that can be correctly matched between the pair of images with transformation. For each interest point, both the location and the detection scale of the interest point are taken into account. The correspondences are defined as the two points x_a and x_b that meet the following two conditions: 1) The scale of x_b is within a factor of $\sqrt{2}$ of the correct scale. 2) The location of x_b is within σ pixels of the correct location, where σ is the detection scale of the keypoint. Because the regions of point neighbourhood of SIFT are denoted by circles centred on the keypoints and with radius proportional to σ, shape information of the interest point neighbourhoods is not considered. The correct scale and location are generated by mapping x_a to x_b using the homography relating the pair of images under consideration. The higher the repeatability rate between two images, the more points can be potentially matched and the better the matching performance is.

4.2.2 Design Parameters for Keypoint Detector

In this section, the performance of the SIFT detector is evaluated, in terms of correspondences and repeatability, which measure the actual and relative number of corresponding regions, respectively. These two parameters indicate to what extent the performance of the SIFT detector is affected by different parameter settings. Besides, because the repeatability only takes into account the location and scale of the detected keypoints but not the similarity between the regions identified by the corresponding keypoints, the influence of different parameter settings is further

evaluated by following a more practical approach, which is to investigate the matching stability by measuring the actual and relative number of correct matches, respectively.

a. Sampling Frequency in Spatial Domain

Prior to deciding the sampling frequency in scale, the amount of prior smoothing σ_p is decided, which is applied to the input image of each octave before building the Gaussian scale space. This parameter is closely related to the sampling frequency in the spatial domain.

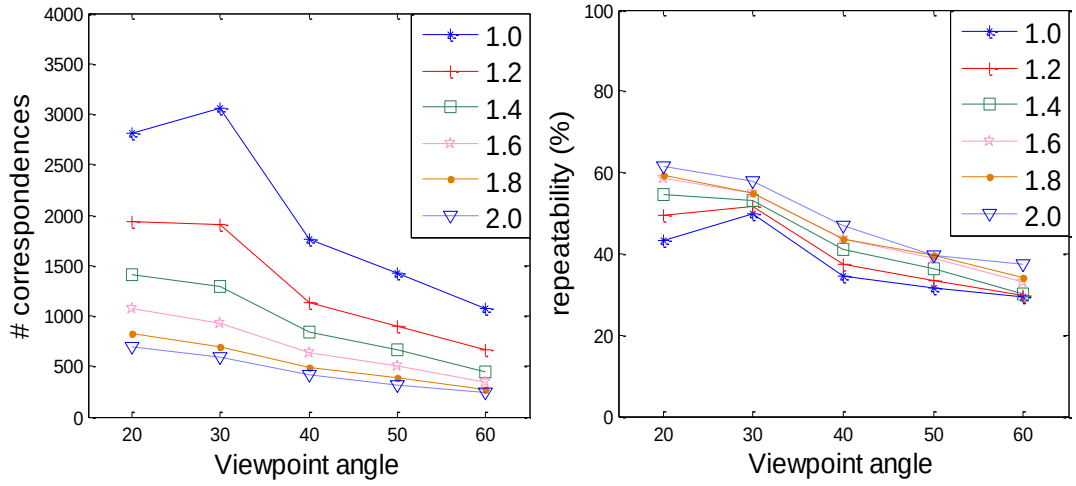


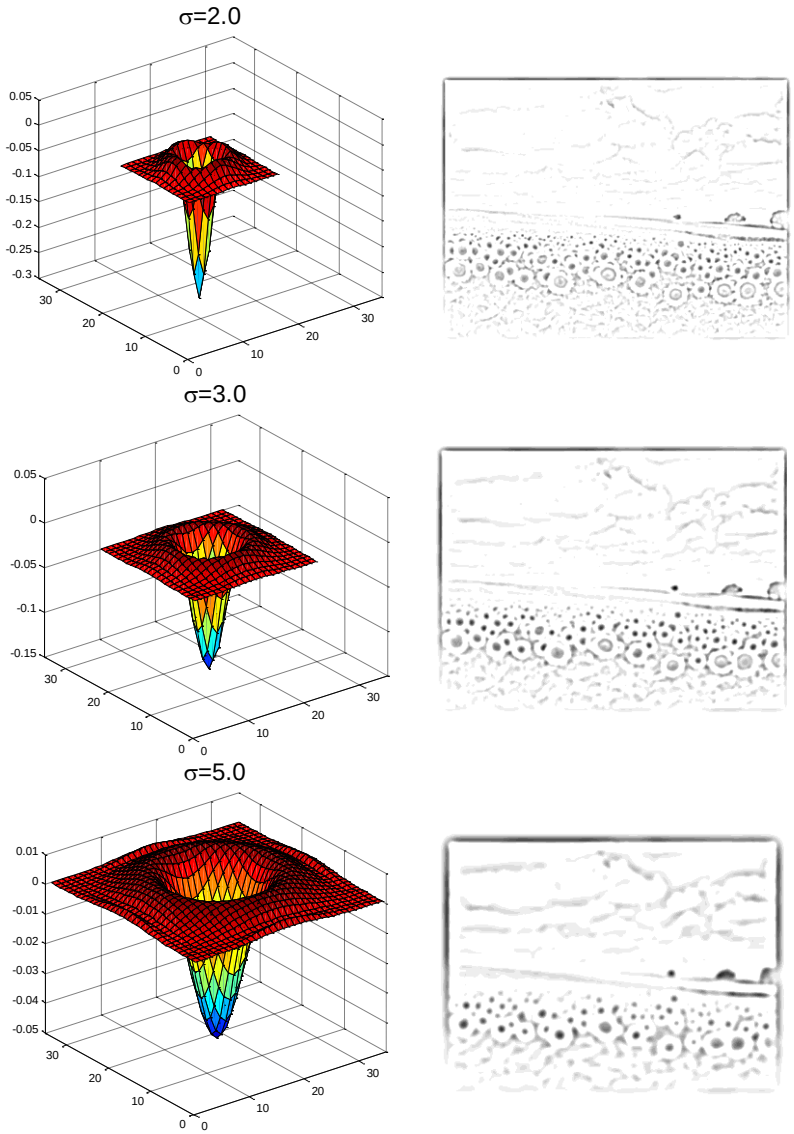
Figure 4-1: Detection performance comparison for different amount of prior smoothing σ_p.

The left image of Figure 4-1 shows that the number of correspondences decreases with the increase of the amount of prior smoothing. This can be understood by the fact that local extrema in DoG scale space can be arbitrarily close together, increasing the amount of prior smoothing actually increases the Gaussian kernel size, which reduces the sampling frequency in spatial domain and hence the number of correspondences as well. In the right image of Figure 4-1, the repeatability increases with the amount of prior smoothing and the ranking of repeatability is opposed to

that of correspondences, indicating that smaller amount of prior smoothing tends to contribute larger number of unstable keypoints that are poorly repeatable. This is because the size of the circular regions featured by the detection scale varies depending on the Gaussian window size, which is closely related to the amount of prior smoothing.

The relationship between Gaussian window size and the detection region is explained using an example shown in Figure 4-2, which illustrates the effect of Gaussian kernel size on the local extrema detection. The detected regions identified by the keypoints detected with smaller σ are on the average smaller, which corresponds to more details of the image contents. These keypoints are regarded as of high locality. The advantage of high locality is that regions identified by these keypoints are less likely to be occluded or suffer from geometric and photometric transformations. However, the disadvantage is that the detected regions contain less information and are less distinguished to survive large transformation. Therefore, the keypoints with high locality are less likely to be repeatedly detected and corrected matched, especially in existence of large image transformation.

(a) Original image.



(b) Left: Difference-of-Gaussian, Right: DoG response. The colour of the DoG response is reversed for clear display.

Figure 4-2: Local extrema as a result of increasing kernel size.

Figure 4-3 shows histograms of region size of the detected keypoints from the same image but with different σ_p, which shows that the overall size of the detection regions rise with σ_p.

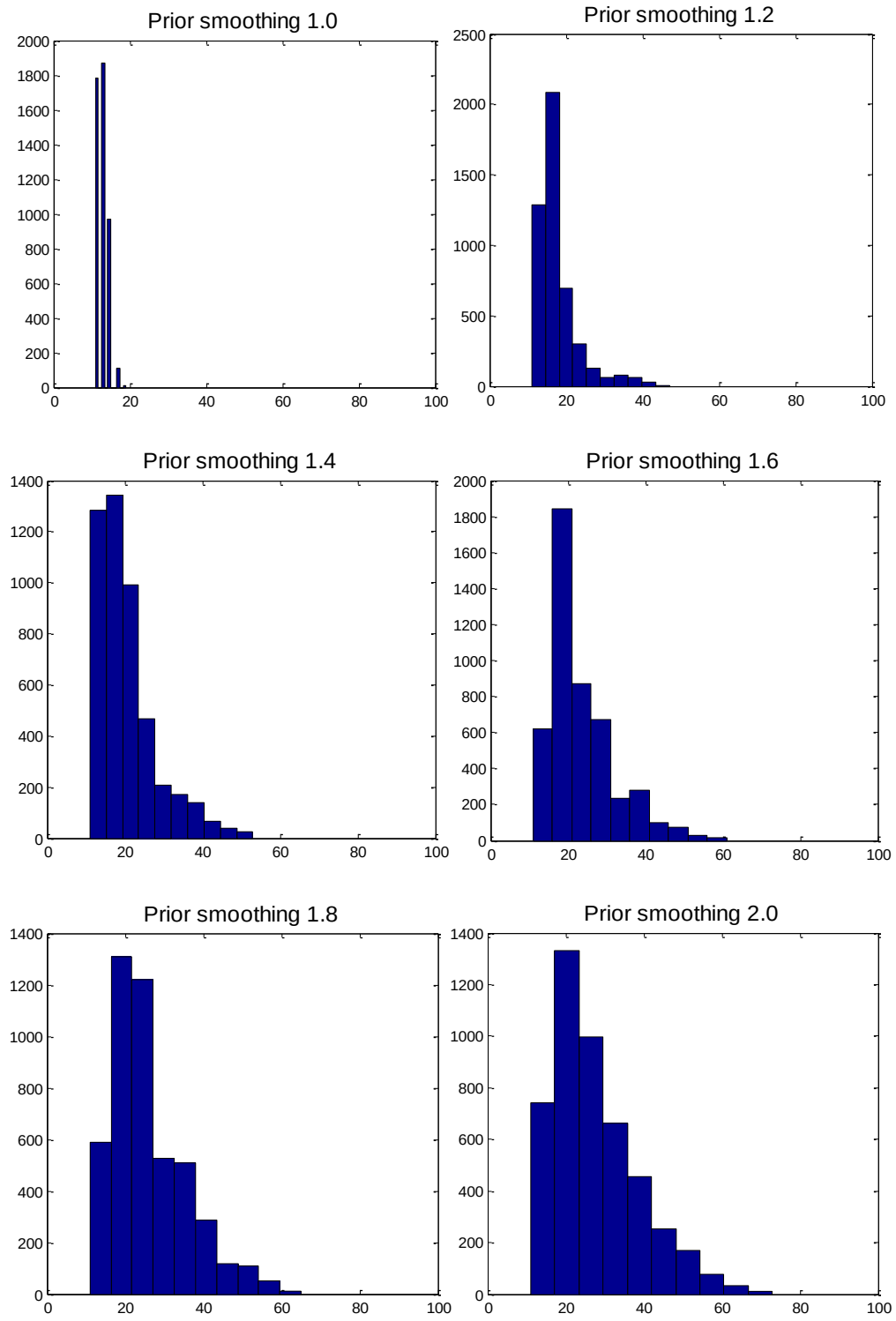


Figure 4-3: Histograms of region size for different amount of prior smoothing σ_p.

Figure 4-4 shows the Gaussian window size as a function of the amount of prior smoothing. The Gaussian window size rises with the amount of prior smoothing, which further increases the computational complexity and time consumption of the hardware design. Therefore, with a trade-off made between the rate of detection, the detection robustness and the hardware efficiency, the prior smoothing is set to σ_p=1.4. Detailed analysis of the size of Gaussian kernel used in the hardware design will be given in Section 4.3.3.

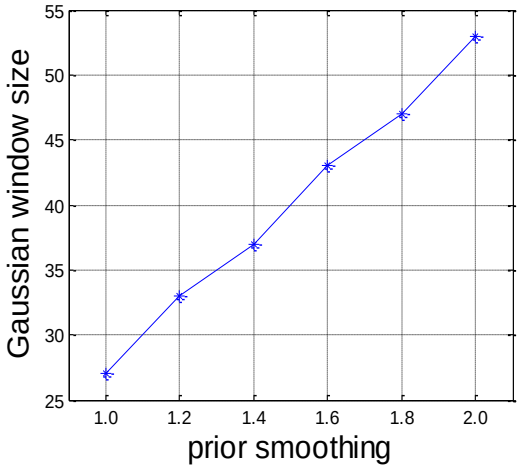
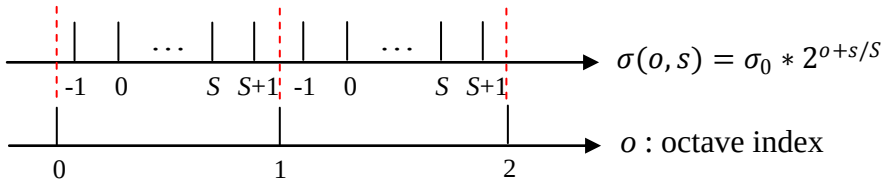


Figure 4-4: Gaussian filter window size as a function of the amount of prior smoothing σ_p.

b. Sampling Frequency in Scale Space

In this section, the sampling frequency in scale space for Gaussian scale space construction is decided by taking into account both the detection performance and hardware efficiency. When building the Gaussian scale space, a limited number of scales and octaves are chosen to represent the continuous scale change in practice. The setting of these two parameters has a great impact on both the detection robustness of the system and the complexity of the hardware design, and hence may differ from one application to another. Generally speaking, higher sampling frequency in scale provides a larger number of keypoints. However, the keypoints are on the average less stable, and hence are less likely to be detected in the transformed image [10]. Besides, the computational cost and memory requirement

also rise with increased sampling frequency of scales. Therefore, the performance of sampling frequency S is evaluated up to three. Figure 4-5 shows the sampling in Gaussian scale space, where each octave consists of $(S+3)$ Gaussian smoothed images. As a result, $(S+2)$ DoG images are produced and keypoints are detected from DoG scale space belonging up to S scales.



S : sampling frequency in scale

σ_0 : prior smoothing factor

s : scale index, in range [-1, S+1]

o : scale index, in range [0, O-1]

Figure 4-5: Sampling of scale for Gaussian scale space construction.

Table 4-1: Gaussian smoothing factors (σ) for different sampling frequency S in scale for Gaussian scale space construction.

Sampling frequency S in scale	$\boldsymbol{S=1}$	$\boldsymbol{S=2}$	$\boldsymbol{S=3}$
σ_0	$\sigma_p * 2$	$\sigma_p * 2^{\frac{1}{2}}$	$\sigma_p * 2^{\frac{1}{3}}$
$s=-1$	$\sigma_0 * 2^{-1}$	$\sigma_0 * 2^{-\frac{1}{2}}$	$\sigma_0 * 2^{-\frac{1}{3}}$
$s=0$	σ_0	σ_0	σ_0
$s=1$	$\sigma_0 * 2$	$\sigma_0 * 2^{\frac{1}{2}}$	$\sigma_0 * 2^{\frac{1}{3}}$
$s=2$	$\sigma_0 * 2^2$	$\sigma_0 * 2$	$\sigma_0 * 2^{\frac{2}{3}}$
$s=3$	-	$\sigma_0 * 2^{\frac{3}{2}}$	$\sigma_0 * 2$
$s=4$	-	-	$\sigma_0 * 2^{\frac{4}{3}}$

Table 4-1 shows the Gaussian smoothing factors for different sampling frequency in scale for Gaussian scale space construction. Instead of doubling the Gaussian smoothing factors, input image to a new octave is generated by down sampling the input image to the previous octave spatially by a factor of two. As a result, the same set of smoothing factors given in Table 4-1 are applied to all octaves, and the computational cost is greatly reduced.

Number of Octaves

Gaussian scale space consists of a limited number of octaves, and each octave is further subdivided into sublevels.

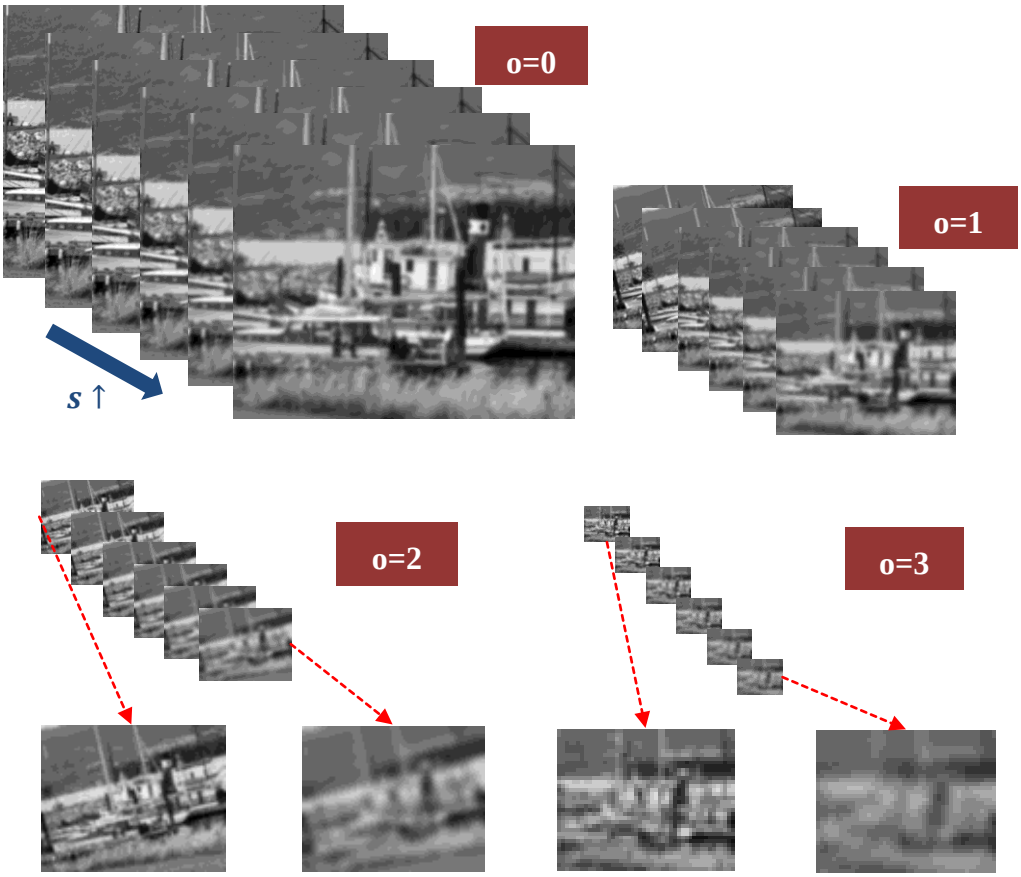


Figure 4-6: Gaussian smoothed images for each of the four octaves ($O = 4$).

Figure 4-6 shows the Gaussian smoothed images of up to four octaves, where the fourth octave produces too small and over-smoothed images, resulting in a low probability of detecting a large number of features with high distinctiveness. Therefore, the performance of up to three octaves is compared. The comparison results are demonstrated by using the boat sequence, as shown in Figure 4-7. In the following experiments, the reference image is always the image of the highest quality and the smallest transformation.

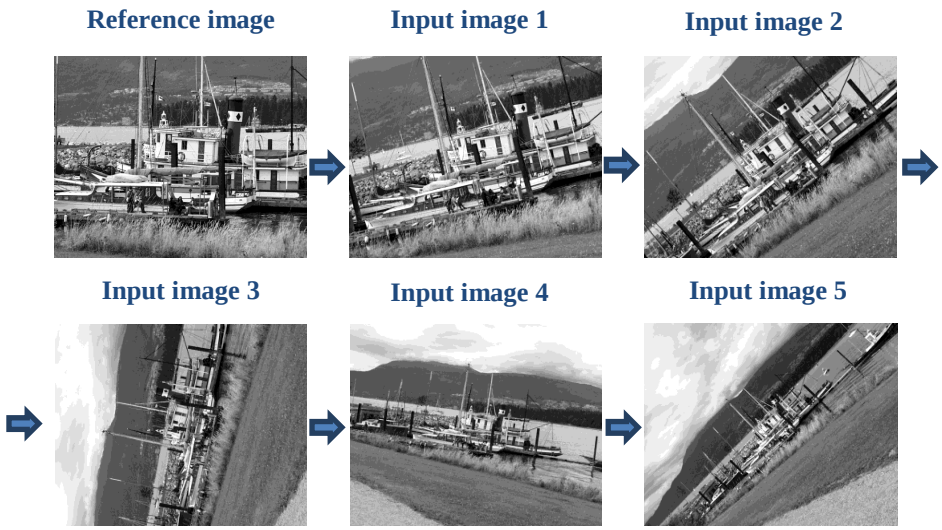


Figure 4-7: The leftmost image in the first row is the reference image, and the arrow indicates the severity of transformation.

The performance is tested for different number of octaves, in terms of detection robustness and matching accuracy. It has been tested that about 82.3% and 16.8% of the total keypoints are detected from the first and second octave, respectively. As shown in the left image of Figure 4-8, less than 1% is from the third octave. The detection result of the reference image is given in the right image of Figure 4-8. Each green dot corresponds to a keypoint detected from the corresponding octave and the number of keypoints detected from each octave is 1550, 321, and 15, respectively. The first two octaves provide a large number of keypoints densely covering the entire image. Keypoints with local region exceeding the image borders have been discarded.

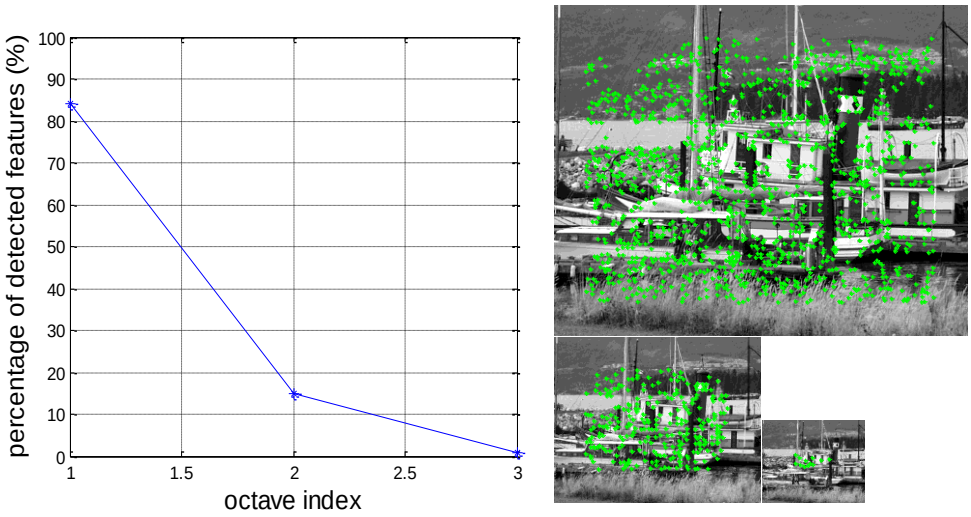


Figure 4-8: Distribution of detected keypoints over octaves.

The detection robustness is evaluated by comparing the correspondences and repeatability. It can be seen from Figure 4-9 that there is an obvious increase in both the number of correspondences and the repeatability when the number of octaves is increased from O=1 to O=2, reflecting that the detection robustness is improved significantly. However, the robustness does not keep improving when more octaves are used, and the detection robustness of O=2 and O=3 are kept at a similar level.

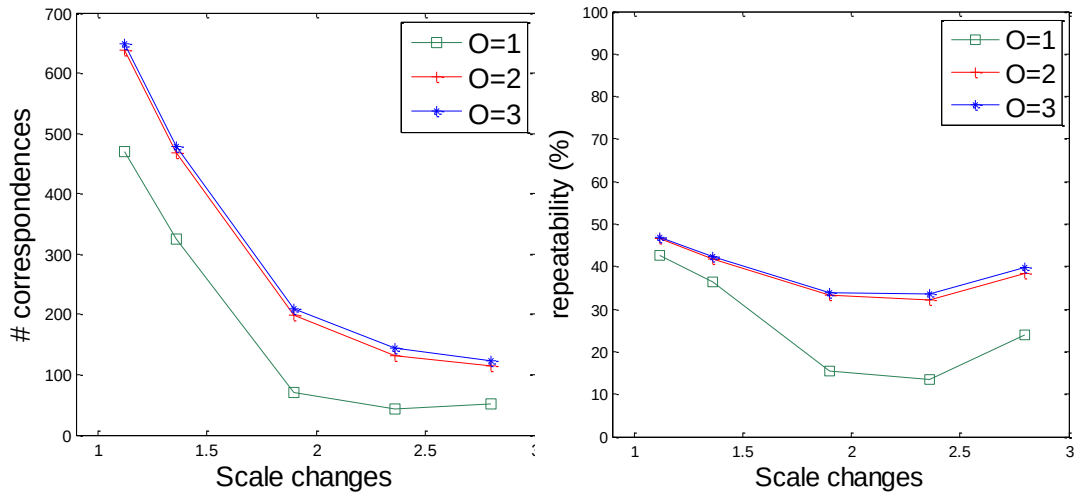


Figure 4-9: Detection performance comparison for different number of octaves. The comparison is performed on the structured scene with scale changes.

Matching performance comparison is shown in Figure 4-10. With the rise of transformation severity, the matching performance of O=2 drops slightly below that of O=3, but it is significantly superior to that of O=1.

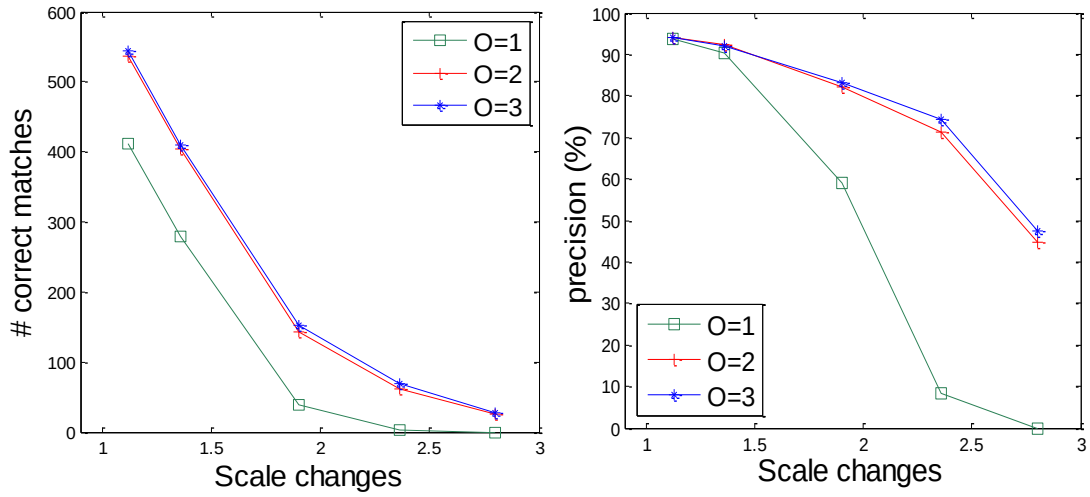


Figure 4-10: Matching performance comparison for different number of octaves.

Because the size of the source images used in the system is 640x480 pixels, and each consecutive octave is the down sampled version of the input image from the previous octave, the third octave is of 160x120 pixels and stands a little chance of detecting a large number of keypoints. Therefore, two octaves (640x480 and 320x240) are chosen to parameterise the design so as to further reduce the memory required to buffer internal calculation results for the third octave.

Number of Scales per Octave

In addition to the number of octaves, the design is also parameterised by the number of scales sampled per octave. Experiments are conducted to determine the sampling frequency S in scale to provide relatively high detection robustness and matching stability.

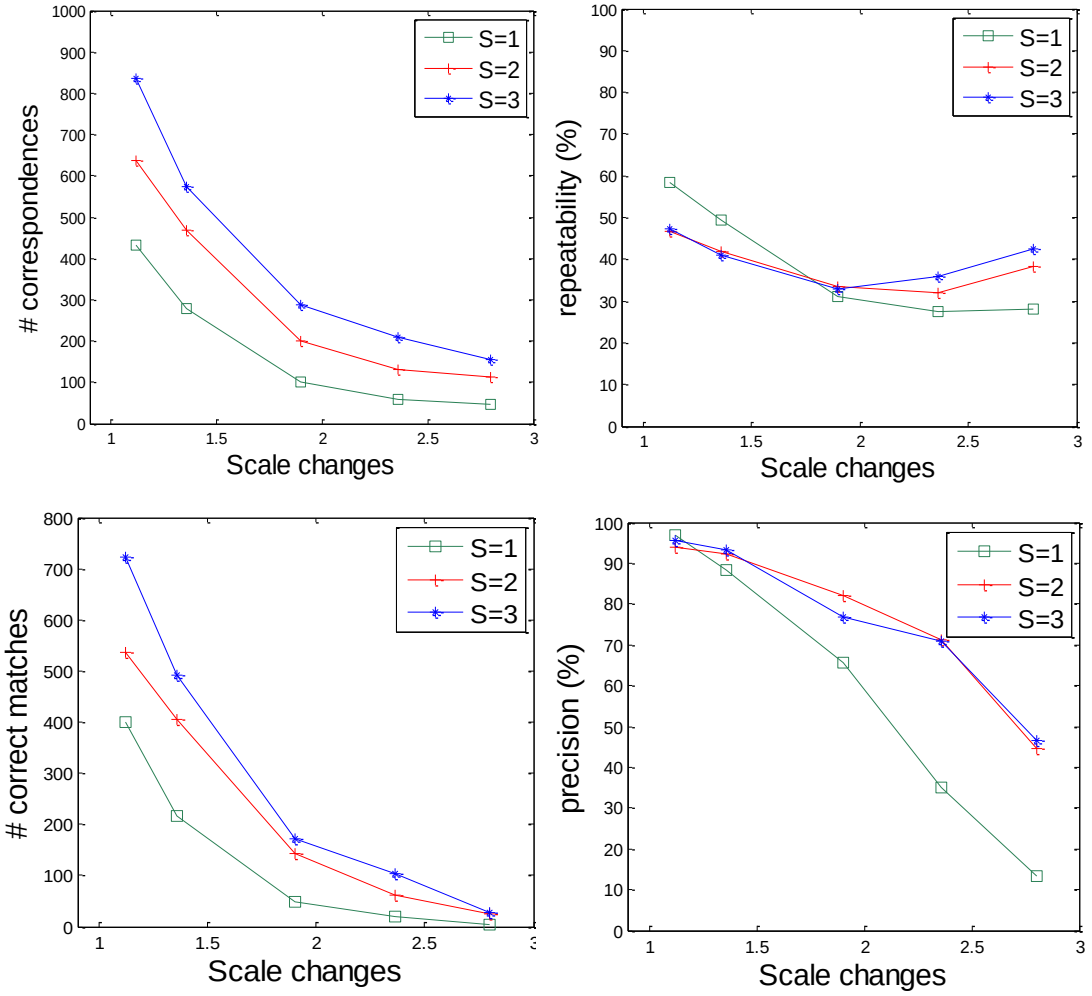


Figure 4-11: Performance comparison for different sampling frequency in scale. The comparison is performed on the structured scene with scale changes.

Evaluation results for the boat sequence (structured scene) and the wall sequence (textured scene) are given in Figure 4-11 and Figure 4-12, respectively. Both the number of correspondences and correct matches decrease significantly with the severity of transformation. In all cases, S=3 performs the best. When the scale changes are small, S=1 obtains the highest repeatability, which is due to the relatively small number of detected interest points in the commonly visible regions of the pair of images under consideration. However, the repeatability of S=1 drops below that of the other two settings as the scaling factor increases, which indicates that the robustness to scale changes of S=1 is relatively low. Although there is a drop in the number of correct matches for S=2 when compared with S=3, the precision

remains at a similar level, which indicates that the overall distinctiveness of the regions detected are not significantly degraded as a result of the reduction in the sampling frequency in scale from S=3 to S=2. But there is an obvious degradation in precision when the sampling frequency is further decreased to S=1.

The difference in the repeatability and precision is increased for the textured scene. It can be seen from Figure 4-12 that both the detection and the matching performance of S=1 are significantly worse than those of the other two settings.

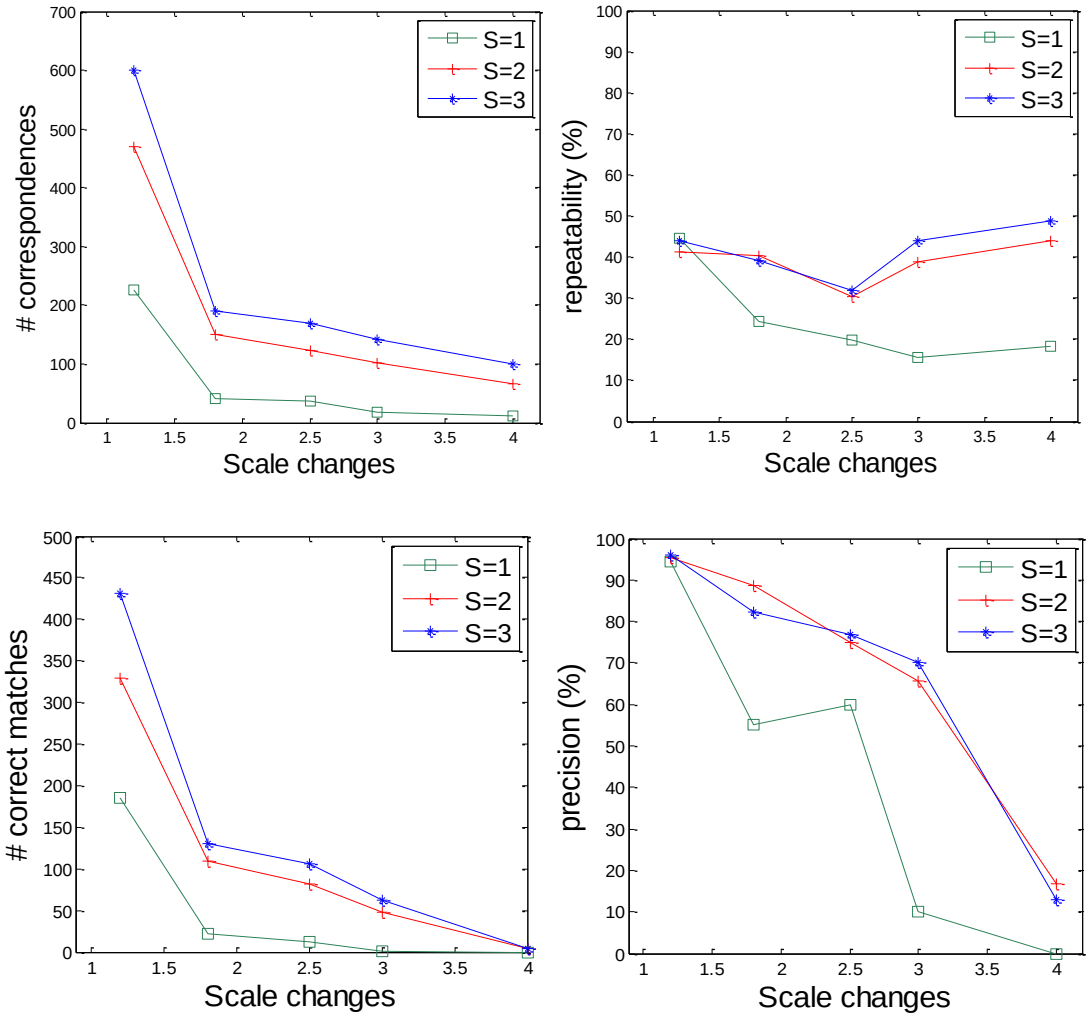


Figure 4-12: Performance comparison for different sampling frequency in scale. The comparison is performed on the textured scene with scale changes.

It can be seen from above experimental results that similar performance is achieved by S=2 and S=3, and S=2 provides a considerable amount of keypoints and correct matches despite of the reduction when compared with S=3. With the rise of the sampling frequency in scale, higher computational cost and larger memory requirement will be introduced into the hardware design accordingly. Therefore, a compromise is made by sampling two scales per octave, which corresponds to five Gaussian smoothed images per octave. As a result, both the detection and matching performance are kept at a relatively high level, while keeping hardware design complexity to the minimum.

c. Effect of Threshold

The effect of threshold on both the detection and matching results are tested to eliminate the possibility that the results reported above are affected by the threshold.

Location Threshold

To eliminate the effect of location threshold on the correspondence determination in the previous experimental results, the performance is compared by varying the threshold. The value was fixed to 1.0 in the previous experiments, which means that a matching location is defined as being within a factor of σ pixels, where σ is the detection scale of the keypoint. It is obvious that more keypoints are qualified as correspondences as the distance threshold is relaxed. However, as shown in Figure 4-13, the overall ranking of each configuration remains virtually the same, indicating that the experimental results are rather indicative than quantitative and are not sensitive to the location threshold.

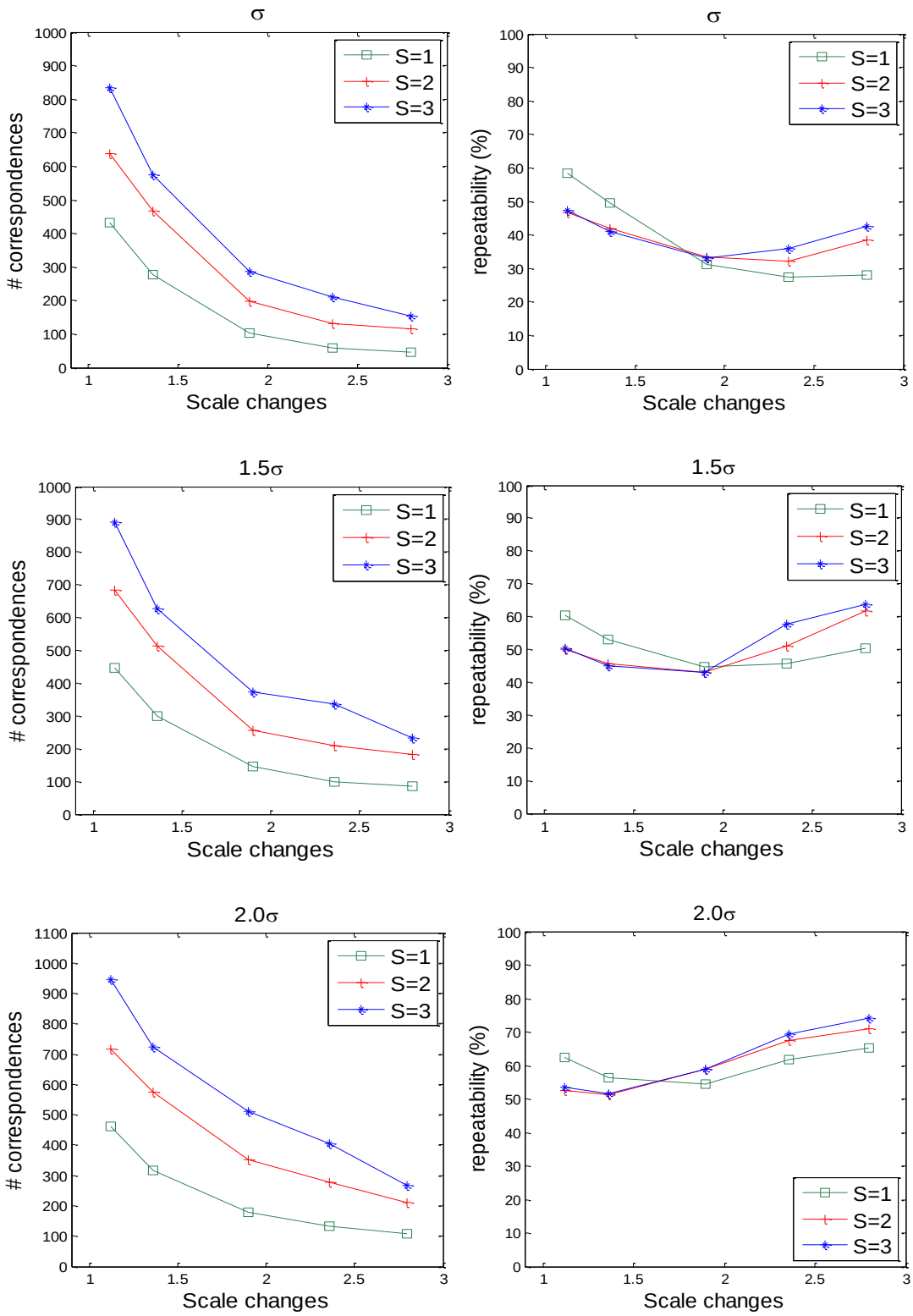


Figure 4-13: Comparison of detection robustness for different distance threshold of keypoint location.

Matching Threshold

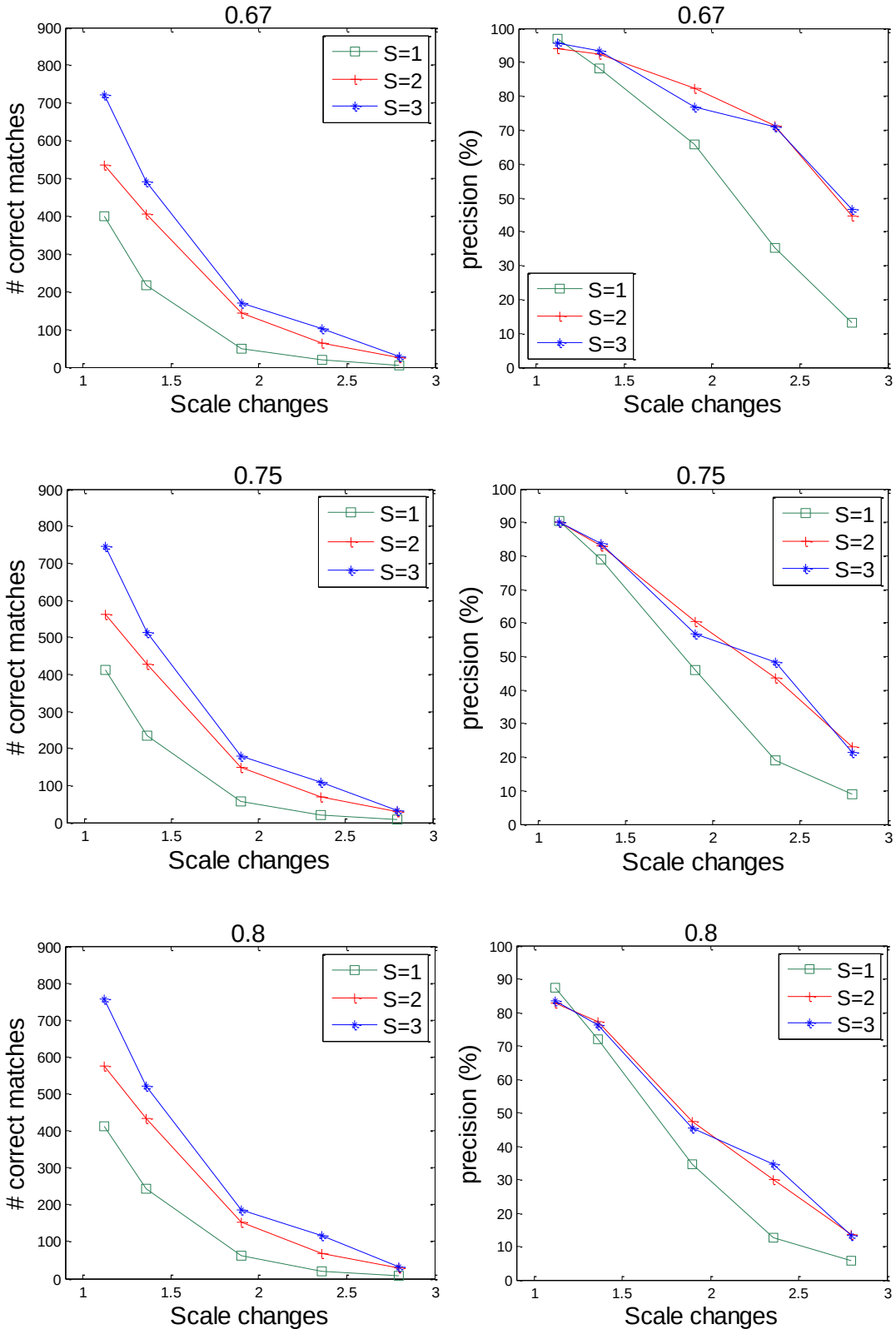


Figure 4-14: Comparison of matching results for different matching threshold.

This section presents the effect of the matching threshold on the determination of correct matches. In the distance ratio based matching, a pair of keypoints is qualified as matched if the ratio of the nearest neighbour to that of the second nearest neighbour is below a predefined threshold. It is obvious that more matches are qualified as correct as the distance threshold is relaxed, but many incorrect matches are qualified as correct as well and the number of incorrect matches increases faster than that of correct matches, and hence the overall precision drops. Figure 4-14 shows that the ranking of each configuration remains virtually the same, which indicates that the experimental results are not sensitive to the choice of matching threshold.

4.2.3 Design Parameters for Descriptor Generation

In this section, the parameters that affect the performance of descriptors are studied. A set of settings is worked out, which is balanced between performance and hardware efficiency, such as the localisation accuracy for descriptor generation, and quantisation precision of principal orientation θ_{po}.

For a given type of descriptor arrangement, there are mainly two factors that affect the computation of descriptors:

1. The localisation accuracy that decides the scale from which the descriptors are computed.
2. The quantisation error of the principal orientation, which corresponds to the accuracy of the principal orientation θ_{po}.

a. Localisation Accuracy

An issue arises on which scale to compute the descriptor for a given keypoint. In the descriptor generation process, each keypoint is first assigned a principal orientation θ_{po} based on the local GMO information within the local region of the keypoint. In the standard SIFT algorithm, θ_{po} is computed based on the smoothed image chosen by the closest scale of the keypoint, so that the orientation assignment is carried out in a scale-invariant manner. The closest scale is the scale image that is nearest to the detection scale to which the keypoint belongs under sub-pixel accuracy, and the size of the local region (r_{local}) is directly proportional to the detection scale (σ). Although

higher precision can be obtained by using the smoothed image patch from the closest scale with size decided by the detection scale, the hardware efficiency is low due to the following two reasons:

1. To reduce the computational complexity and the processing time of descriptor generation process, GMOs are typically computed in parallel with feature detection and buffered for fast indexing for descriptor generation. However, by computing the descriptors based on the closest scale, GMOs of all possible scales have to be computed and buffered, resulting in a significant memory usage.
2. Prior to generating the descriptor, the gradient magnitude of all pixels within each sub-region has to be assigned a weight by applying a Gaussian weighting function with standard deviation of σ_{DAISY}. As has been discussed in Chapter 3, the standard deviation σ_{DAISY} is proportional to the radius r_{local} of the local region, which is further proportional to the detection scale σ of the keypoint. This requires the Gaussian coefficients to be computed during descriptor generation process and hence is ineffective.

To improve the hardware efficiency, a trade-off should be made between descriptor performance and hardware efficiency. The issue of localisation accuracy is analysed in the following two aspects to deal with the above mentioned two factors that affect the hardware efficiency.

1. Calculate the descriptor based on the pre-defined scale instead of the closest scale.
2. Decide the size of local region r_{local} based on the standard deviation of pre-defined scale instead of the detection scale σ of the keypoint.

Scale Selection for Descriptor Computation

Experiments are first conducted to see how the precision of θ_{po} varies with the descriptor computed on pre-defined scale instead of the closest scale. Figure 4-15 shows the probability distribution of Δs, where Δs is the distance from the refined location under sub-pixel accuracy (detection scale) to the origin (pre-defined scale)

in s direction. According to the experiments, only an average of 2.5% of total keypoints is refined closer to an adjacent scale.

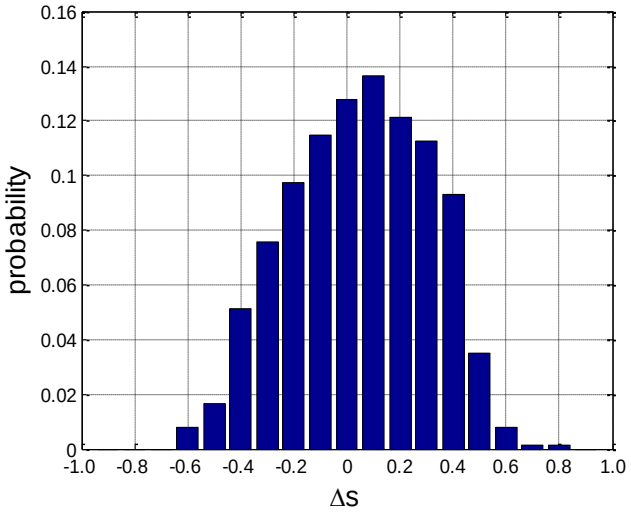


Figure 4-15: Probability distribution of Δs.

A detailed analysis is performed on a pair of images, namely the first and third image in the first row of the boat sequence given in Figure 4-7. Figure 4-16 shows the scale selection for keypoints, which shows that the closest scale of most keypoints are consistent with that of the pre-defined scale, despite of a small number of outliers. In this example, 28 of the total 1038 detected keypoints are refined to an adjacent scale.

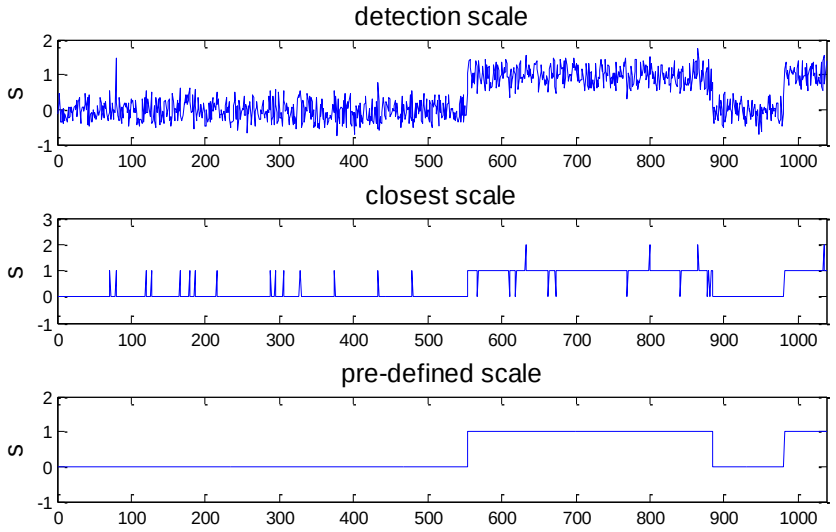


Figure 4-16: The top image shows the detection scale of keypoints. The middle image shows the closest scales. The bottom image is for the pre-defined scales.

Experiments are conducted to check the effect of scale selection on the orientation assignment. As shown in Figure 4-17 , of the total 28 keypoints with shift in scale, 9 have the peak shifted into adjacent bins and 1 has a significant shift in peak, whereas the rest remains unchanged even with a shift in scale.

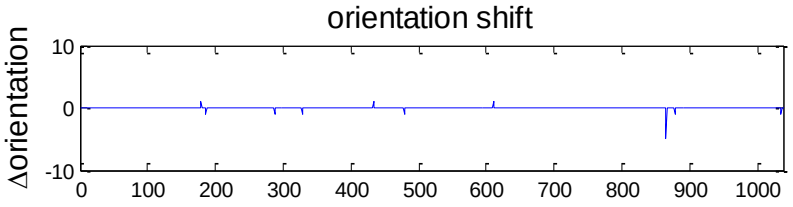


Figure 4-17: The top image shows the shift in scale. The bottom image shows the shift in θ_{po}.

Figure 4-18 shows an example where the magnitudes of the peak and that of the adjacent bin are of high similarity. In this example, θ_{po} is shifted to an adjacent bin as a result of the shift in scale. Since the 36-bin histogram will be further interpolated into 8-bin histograms, the effect on the histogram arrangement will be reduced by the interpolation process.

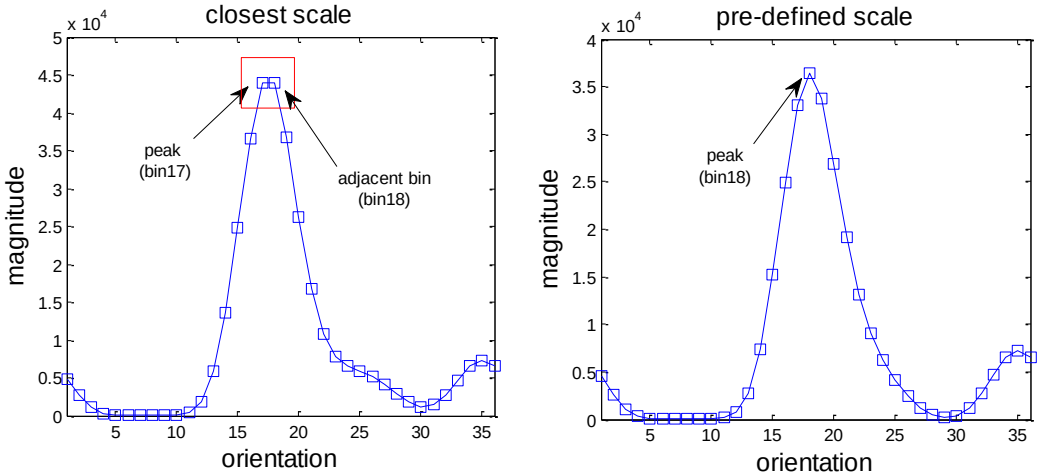


Figure 4-18: The 36-bin histogram for θ_{po} calculation with the peak shifts to an adjacent bin as a result of the shift in scale.

Figure 4-19 gives an example where the peak remains the same and is not affected by the scale selection, whereas the peak in Figure 4-20 is shifted from bin12 to bin31 due to the fact that there are multiples peaks of similar magnitude. This can be compensated by creating keypoints for any local peak that is within 80% of the highest peak of the 36-bin histogram, with which there might be multiple keypoints created at the same location but with different orientation. As a result, the orientation assignment is not significantly affected by using pre-defined scale when compared with the results based on the closest scale.

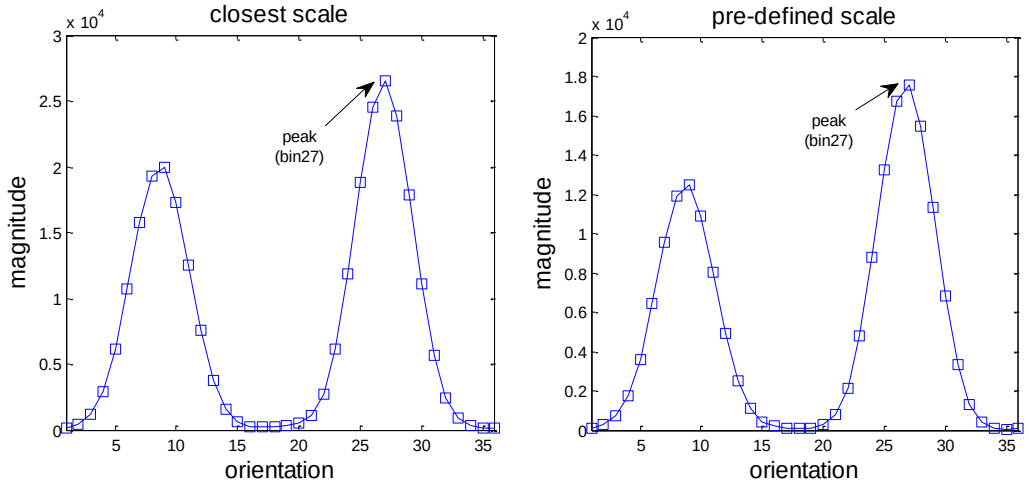


Figure 4-19: The 36-bin histogram for θ_{po} calculation with no shift in peak.

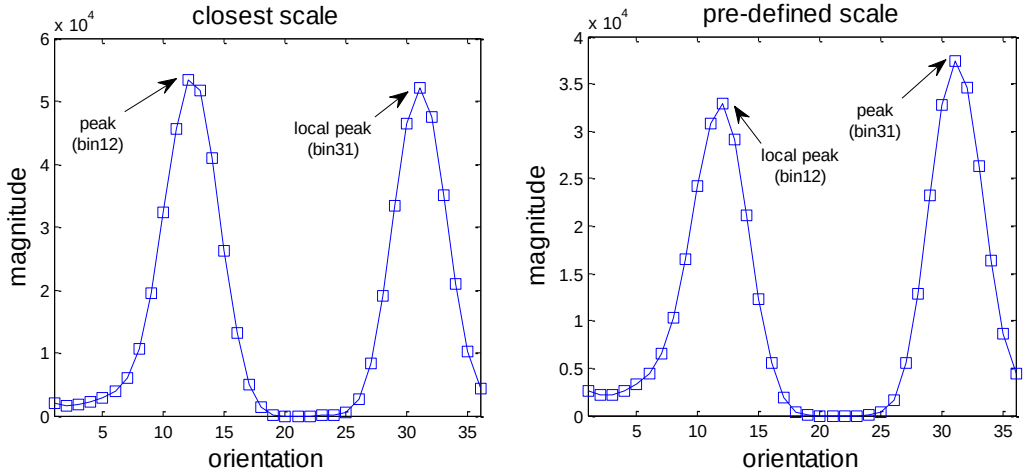


Figure 4-20: The 36-bin histogram for θ_{po} calculation with a significant shift in peak as a result of the shift in scale.

Scale Selection for Region Size Determination

The region size (r_{local}) determination based on the detection scale requires the Gaussian coefficients to be computed in real-time, which increases the computational complexity and the processing time of descriptor generation. To improve the hardware efficiency, r_{local} is determined by using the pre-defined scale. However, the error in scale value affects the descriptor computation as the operation requires the selection of an image patch around the point which is proportional to the selected scale value. To separate the effect of scale selection for GMOs from the scale selection for region size r_{local}, GMOs are computed from the closest scale in this experiment.

Figure 4-21 shows the experimental results with r_{local} proportional to the detection scale and the pre-defined scale, respectively. The two curves are separated by a small gap.

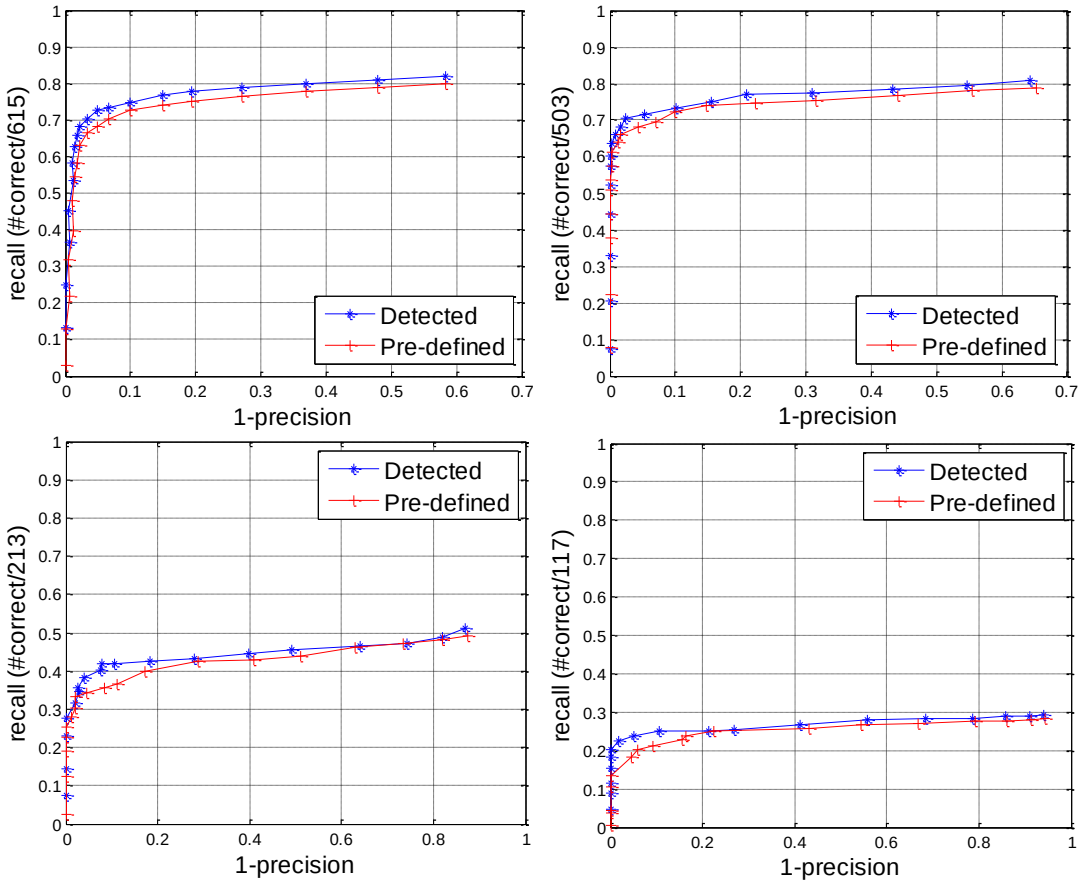


Figure 4-21: Matching performance comparison between the detection scale and the pre-defined scale.

An example of the overall matching performance comparison is shown in Figure 4-22, which shows that the last two curves are nearly identical, confirming that computing descriptor based on the pre-defined scales does not affect the descriptor performance significantly.

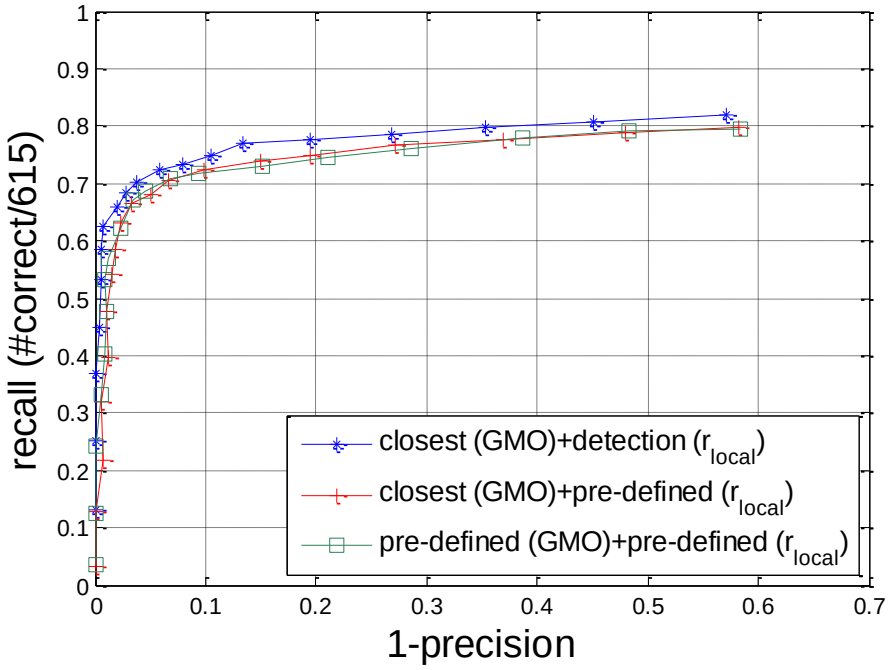


Figure 4-22: The overall matching performance comparison for localisation accuracy.

With descriptors computed based on the pre-selected scales instead of the closest scales, the necessity of computing and buffering GMOs of all possible scales is avoided. With the design parameterised by two octaves of five scales each, it is possible to have features belonging up to two pre-defined scales, and hence scale space information of two smoothed images (per octave) is sufficient. As a result, computing GMOs within the neighbourhood of the keypoints during the descriptor generation process is replaced by indexing into the buffer holding GMOs from the pre-defined scales, which reduces the computational complexity and the processing time while keeping the memory usage to a minimum level.

With r_{local} determined by the pre-defined scale, r_{local} is directly proportional to the pre-defined scale, and hence σ_{DAISY} is known and the Gaussian coefficients can be computed offline and pre-loaded onto an LUT for fast indexing. As a result, the computational complexity of descriptor generation can be further reduced at the cost of a slight drop in the matching performance.

Table 4-2 lists the parameters for SRI-DAISY arrangement with keypoints belonging up to two pre-defined scales.

Table 4-2: Parameters for SRI-DAISY descriptor arrangement.

σ	r/r_{local}	F_{region}	r_{local}	R	r
1.9799	0.35	4.0	20	13	7
2.8			32	21	10

b. Quantisation Precision of Principal Orientation

Because the descriptor is arranged relative to the principal orientation, the accuracy of the principal orientation has a large impact on the rotation invariance of the descriptor. Each feature is assigned a principal orientation that corresponds to the largest bin in the n-bin histogram of the neighbouring region, where n is the number of orientations covering 360^o. Experiments are conducted on the boat sequence with in-plane rotation and scale changes and the results are given in Figure 4-23. In general, the rotation invariance is enhanced with larger number of orientation bins. With 4 bins covering 360^o, the number of correct matches drops significantly even with a slight rotation angle. The number of correct matches is improved significantly when the orientation is increased from 4 to 16, but it shows little change beyond that value. Virtually the same performance is obtained by 36 and 72 directions, indicating that a larger number of orientation bins does not keep improving the rotation invariance and hence is unnecessary. Therefore, the 360 degree range of orientations is quantised to 36 directions.

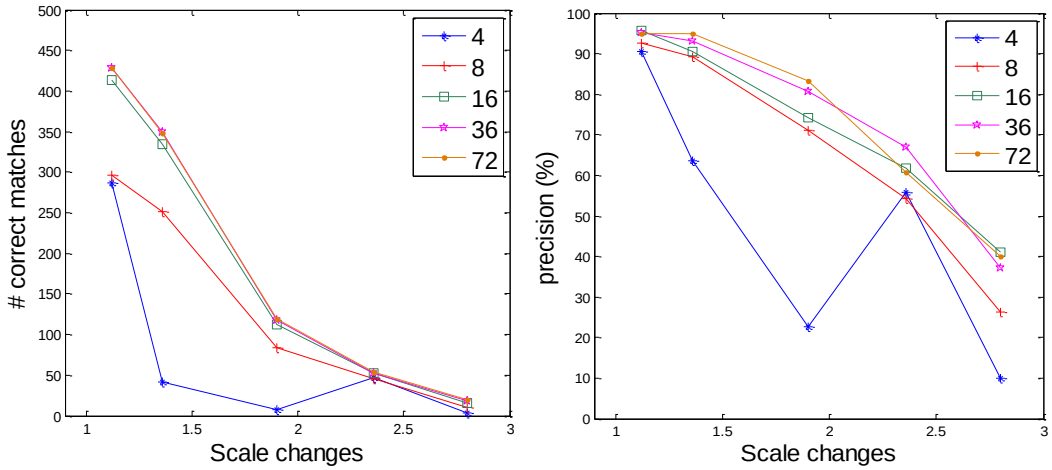


Figure 4-23: Matching results for different quantisation precision of the principal orientation.

4.3 Error Analysis

In this section, experiments are conducted to formulate an appropriate fixed-point model for the SIFT processing core. The simulation results are presented to see the functionality and accuracy of the fixed-point based hardware design. The MATLAB model with floating-point accuracy is used as a reference.

4.3.1 Computational Complexity

As has been introduced in Chapter 2, the first stage of the feature detection module is Gaussian smooth. In a digital form, Equation (4.2) can be written as (4.3).

$$L(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-(x^2+y^2)/2\sigma^2} * I(x,y) \tag{4.2}$$

$$L(x,y) = \sum_{i=-\frac{k}{2}}^{\frac{k}{2}} \sum_{j=-\frac{k}{2}}^{\frac{k}{2}} G(i,j) * I(x+i, y+j) \tag{4.3}$$

where $G(i,j)$ denotes Gaussian kernel coefficient, k is the size of the Gaussian kernel applied, and $I(x,y)$ is the image patch to which the Gaussian kernel is applied.

As can be seen from Equation (4.3), the number of calculations increases non-linearly with the size of Gaussian kernel k. Without any optimisation, smoothing a pixel requires k^2 multiplications and $(k^2 - 1)$ addition operations. This number then needs to be multiplied by the number of pixels to be processed from the input image. The number of operations is then multiplied by the number of smoothed images within each octave, since Gaussian scale space consists of a number of smoothed images produced from the convolution of Gaussian kernel of different standard deviations with the input image. If there are multiple octaves within the Gaussian scale space, the number will be further increased. Therefore, the computational complexity of Gaussian smooth process is decided by the size of input image, the number of scales within each octave, and the number of octaves.

The decision of appropriate word length of fixed-point arithmetic is important because it affects the resource usage and performance of the system. The word length of the Gaussian kernels applied for Gaussian scale space construction affects the implementation efficiency of the system. Increasing the word length provides smoothed images with higher accuracy, but the system becomes more complicated as the number of bits increases after each processing step, which will be discussed later. Therefore, word length of Gaussian coefficients has to be studied and bit-truncation is necessary at specific stages in the calculation process in order to reduce complexity overheads.

The stability checking process also affects the throughput of feature detection module, because it requires the location of the detected extrema to be repeatedly refined and each refinement process involves matrix inversion that is expensive and time consuming to be implemented on hardware devices. Besides, the data dependency between location refinement process and low contrast removal prevents these two processes from being implemented in parallel, which limits the throughput.

A large memory is required to buffer GMOs (Gradient Magnitude and Orientations) for descriptor generation and the descriptors. Analysis has to be performed to trade-off between memory and descriptor precision.

Table 4-3 summarises the error analysis performed in this section.

Table 4-3 List of error analysis presented in this section.

Module	Error Analysis
Feature Detection	Maximum Gaussian kernel size
	Word length of Gaussian kernel coefficients
	LSBs truncation on Gaussian filtered images
	Word length of DoG values
	Maximum number of iterations for localisation refinement of keypoints
	Approximation on low contrast removal
Descriptor Generation	Precision of the Principal Orientation Calculation
	Quantisation Error of Feature Descriptors

4.3.2 Simulation Scheme for Feature Detection

As shown in Figure 4-24, three comparisons are performed to evaluate the performance of the fixed-point based hardware design for feature detection module. The experiments are conducted on the boat sequence.

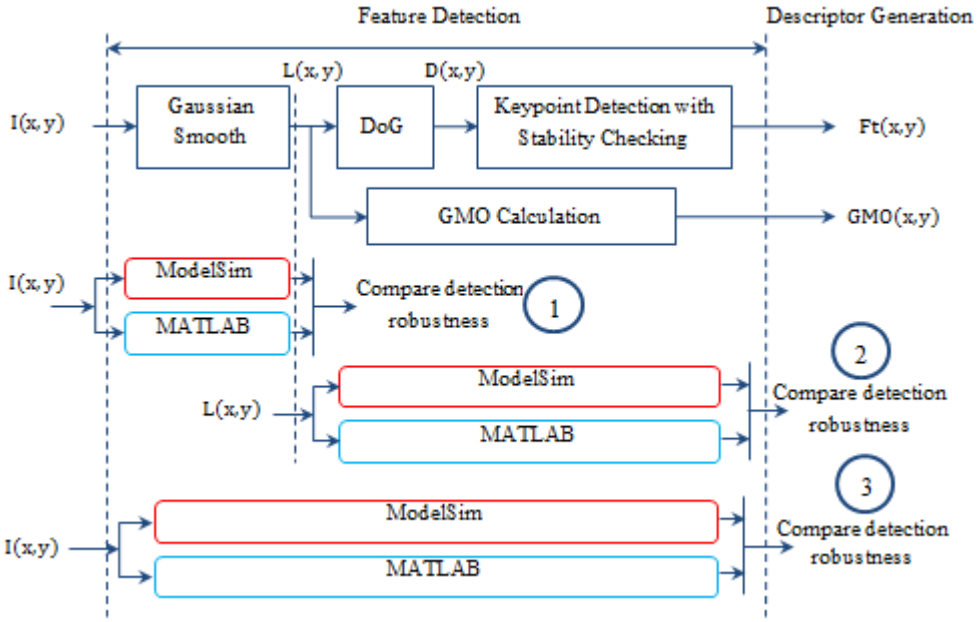


Figure 4-24: Three comparisons to evaluate the processing accuracy of feature detection.

4.3.3 Error of Gaussian Scale Space Construction

The error of the scale space construction refers to the comparison "Step 1" as shown in Figure 4-24, which is evaluated in two aspects: 1) the size of the discrete Gaussian filter window, 2) the quantisation of Gaussian coefficient using fixed-point accuracy.

a. Gaussian Kernel Size Error

The coefficients of discrete 1D Gaussian kernel of size $k = (2j + 1)$ can be calculated by using Equation (4.4).

$$G_i = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{i^2}{2\sigma^2}} \tag{4.4}$$

where j decides the radius of the Gaussian window, and i is integer in range -j to j.

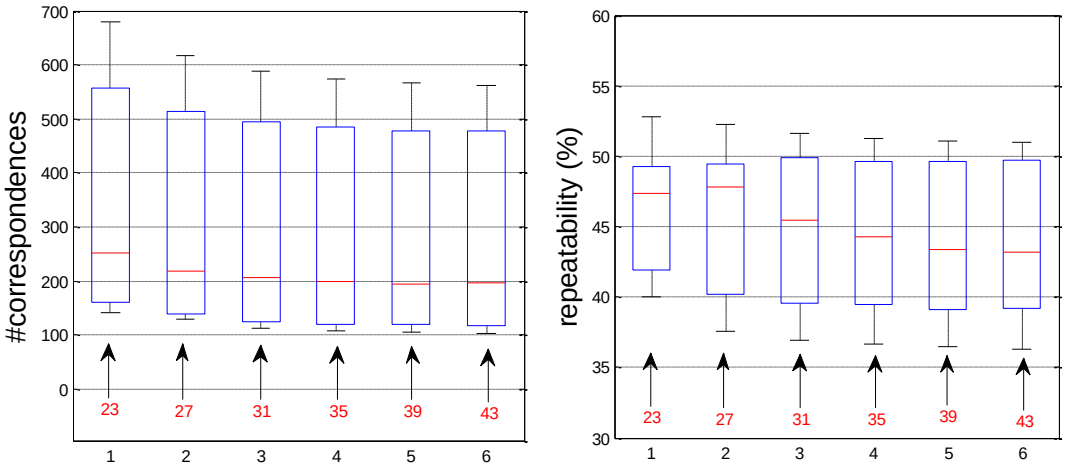


Figure 4-25: Detection performance as a function of gradually increasing Gaussian kernel size.

The effect of Gaussian kernel size is determined by looking into both the detection and matching performance. Figure 4-25 gives the experimental results from a pair of images, which shows how the performance varies with the size of Gaussian kernel. k_G=43 corresponds to Lowe's software model. Figure 4-25 shows that detectors with

smaller kernel sizes suffers from over-detection when compared with the reference, which potentially increases the memory requirement and processing time of the descriptor generation module.

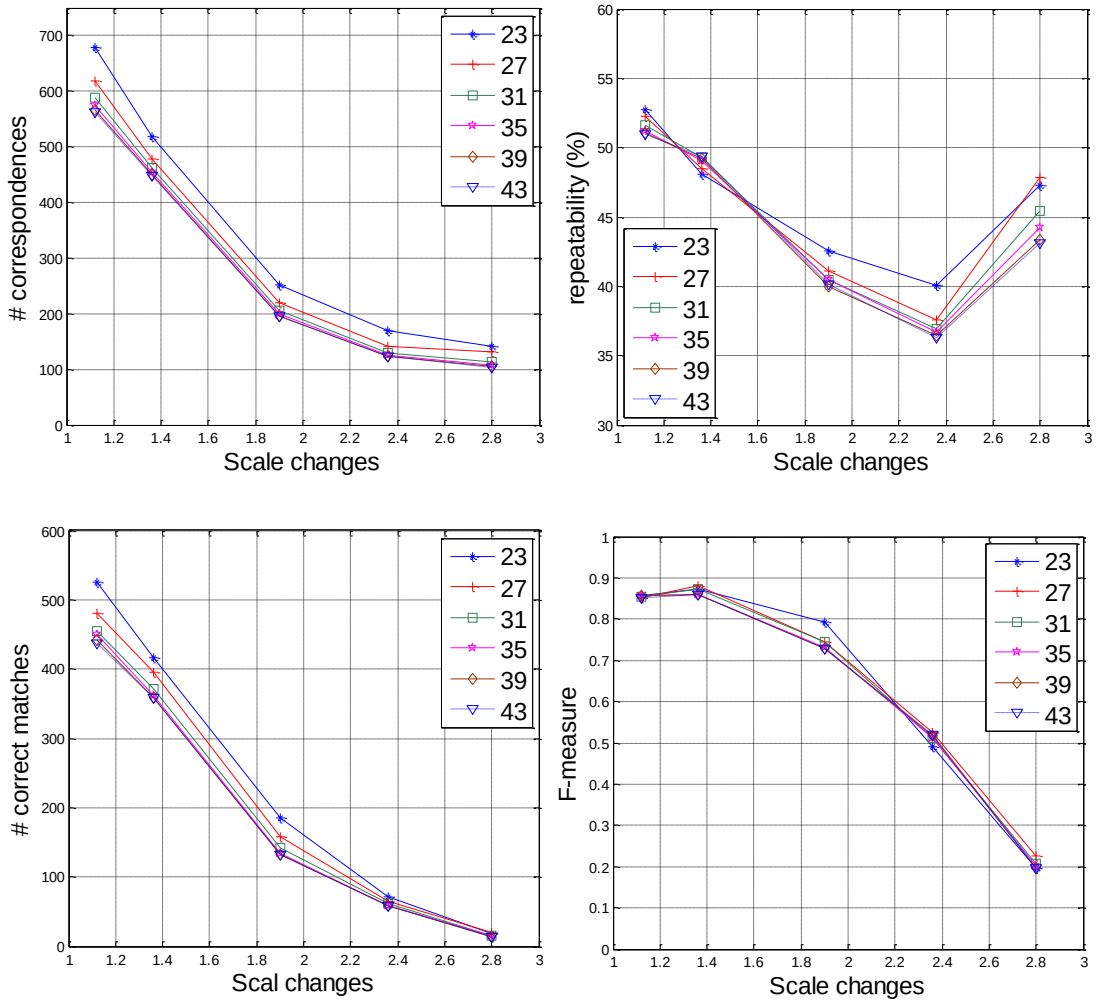


Figure 4-26: Detection and matching performance for different Gaussian kernel sizes. k_G=43 corresponds to Lowe's software model.

Figure 4-26 gives the experimental results from a set of images with gradually increasing transformation, which shows how the detection and matching performance varies with the severity of transformation for k_G of different values. The difference in repeatability increases with the severity of transformation, and the performance of k_G larger than 27 is similar to the reference. Besides, although the

relative ranking of the number of correspondences remains virtually the same with the severity of transformation, the number of correct matches drops faster for smaller Gaussian kernels. This can be understood by the fact that the radius of local region is proportional to the size of the corresponding Gaussian kernel k_G, and larger regions typically contain more information and hence are more discriminative to survive large transformation. Example local regions identified by keypoints detected with k_G=23 and k_G=43 are given in Figure 4-27. The red and green circles represent the local regions identified by keypoints detected using k_G=23 and k_G=43, respectively. The size of local regions represented by green circles is on the average larger than those represented by red ones. However, the number of local regions of the former is smaller than that of the latter, which agrees with the performance shown in Figure 4-26.

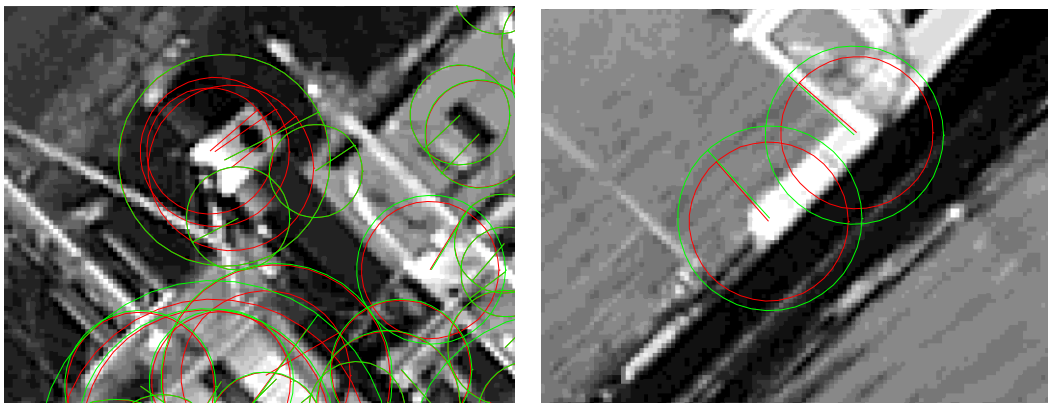


Figure 4-27: The red and green circles represent the local regions represented by keypoints detected with k_G=23 and k_G=43, respectively.

To make use of the parallel processing property of the FPGA, Gaussian kernels of different sizes are applied to the source image concurrently for Gaussian scale space construction, and the processing time is directly related to the size (k_G) of the largest Gaussian kernel applied. To keep relatively high accuracy while achieving the target system throughput (60 fps), k_G is set to 31, with which both the detection and matching performance are kept at a similar level to that of Lowe's software model. Because the size of the other four Gaussian kernels does not have effect on the

system throughput, they are kept at the same level to that of Lowe's software model. As a result, the Gaussian kernels are of size 13, 17, 25, 29, and 31, respectively.

b. Fixed-point Error

There are two basic operations in the Gaussian filter process: addition and multiplication, which can be implemented in either fixed-point or floating-point format. This design uses the two's complement fixed-point arithmetic, as shown in Figure 4-28. The data consists of a sign bit, an integer part, and a fractional part. Generally speaking, the floating-point implementation provides a larger dynamic range and hence higher calculation accuracy, but usage of floating-point arithmetic is expensive on hardware devices and leads to inefficient designs especially for FPGA implementation. On the other hand, the fixed-point implementation consumes less hardware resources and offers higher processing speed, and hence more efficient hardware designs. However, using fixed-point arithmetic can result in a reduction in the accuracy if it is not carefully designed. This section formulates an appropriate fixed-point representation for Gaussian kernel coefficients that maintains calculation accuracy similar to the floating-point implementation.

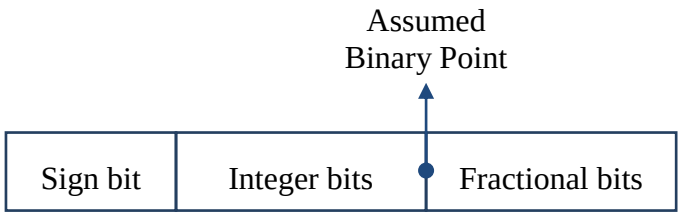


Figure 4-28: The two's complement fixed-point representation.

Word length of Gaussian Filter Coefficients

The first stage of the SIFT processing core is the Gaussian filter. The word length of the Gaussian coefficients affects the complexity of the core because the dynamic range of the intermediate calculation results keeps increasing step by step, which further increases the resource usage of the design. For example, a full length $m \times n$ multiplier yields an output of $(m + n)$bits. To deal with the bit-increasing issue, a

proper word length has to be selected for Gaussian coefficients, and data truncation is performed at the output of calculation steps where necessary. The number of correspondences and the repeatability are checked with gradually increasing fractional bits for Gaussian coefficients. It should be noticed that the Gaussian kernel has to be normalised after scaling up the coefficients by a factor so as not to change the average grey level of the image. The detection performance is tested as a result of the limited precision of Gaussian kernel coefficients. In this experiment, the maximum Gaussian kernel size is set to k_G=31. The output is indicative rather than quantitative, and the Gaussian kernel size does not affect the relative ranking of the outputs. This idea applies to all the experiments in the following sections.

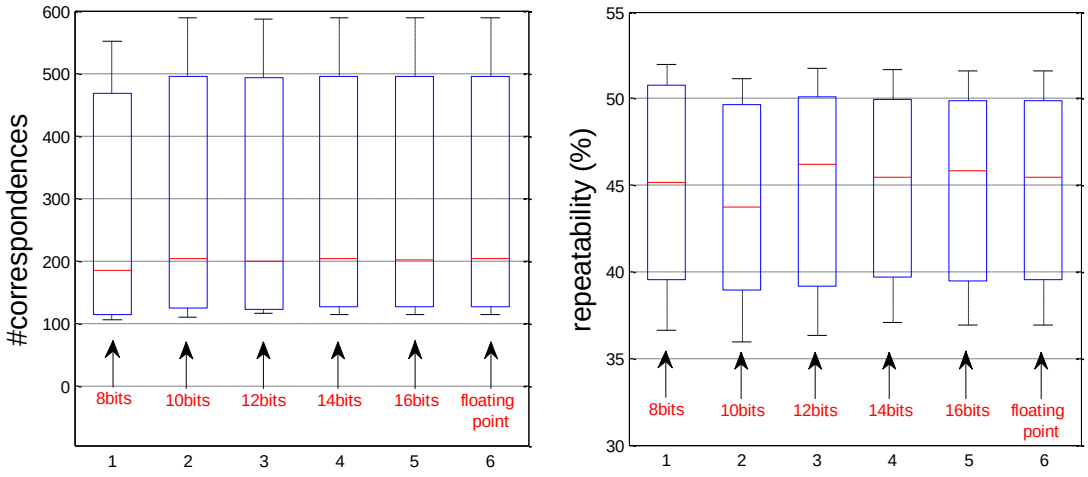


Figure 4-29: Correspondences and repeatability as a function of fractional bits.

Figure 4-29 shows that when the fractional bits are gradually increased from 8 to 16, both the number of correspondences and the repeatability converge to those of the floating-point model. The detection performance becomes rather stable when the coefficient is represented by more than 12 bits. To save hardware resources while preserving relatively high robustness of the feature detector, each Gaussian kernel coefficient is represented by 14 bits.

Data Truncation

Data truncation is necessary in that the hardware resource usage for implementing operations increases accordingly as a result of the increment of the word length after each computation step, such as adders and multipliers. Data truncation can be performed on either the Most Significant Bit (MSB) [62] or Least Significant Bit (LSB) [63] to minimise the hardware implementation cost. When the dynamic range of the signals being processed is much smaller than the peak value for the bit-width used, MSB truncation can be performed to reduce the dynamic range while preserving high accuracy. On the other hand, the LSB truncation keeps the original dynamic range at the expense of accuracy.

One major task while working with the fixed-point arithmetic is to prevent overflow and incorrect results, which occurs when a result may not fit into the reserved word length. To prevent overflow, experiments are conducted to determine the maximum word length of Gaussian filtered results while keeping the area usage to a minimum level. Table 4-4 shows the theoretical maximum word length at each stage of the Gaussian filter process. In each 1D Gaussian filter process, the word length of input signals is extended to avoid overflow, which leads to the final output of 46 bits theoretically.

Table 4-4: Theoretical maximum word length of Gaussian filter process.

Input/output	**Maximum Word length (bits)**
Input Pixel (**I**)	8
Input 1D Gaussian filter coefficient (**G**)	14
Output of 1D vertical filter	27
Final Filtered Pixel (**L**)	46

However, the practical maximum word length can be different. The Virtex-6 FPGA device provides advanced DSP48E1 slice and each supports multiplication with input data of either 18x25 signed or 17x24 unsigned [64]. Larger multipliers are built by

assembling these embedded multipliers. With the input of 1D Gaussian filter in the horizontal direction truncated from 27 bits to 22 bits on the MSB, embedded multipliers are saved and the final output is reduced from 46 bits to 36 bits, as shown in Table 4-5.

Table 4-5: Maximum word length of Gaussian smooth process from real data sources.

Input/output	**Maximum Word length (bits)**
Input Pixel (**I**)	8
Input 1D Gaussian filter coefficient (**G**)	14
Output of 1D vertical filter	22
Final Filtered Pixel (**L**)	36

To further reduce the requirement for hardware resources, LSB truncation is performed at the output of Gaussian filter process. Figure 4-30 shows the Mean Square Error (MSE) of Gaussian filter with LSB truncation at the output. In this section, the MSE is used to measure the difference between the values obtained with and without data truncation. The MSE is defined by Equation (4.5).

$$MSE = \frac{\sum_0^N [L_{orig}(i) - L_{trunc}(i)]^2}{N} \tag{4.5}$$

where L_{trunc} and L_{orig} are the intensity value of Gaussian filtered pixels with and without truncation, respectively, N is the number of pixels involved in the computation of MSE. Smaller values of MSE indicate that L_{trunc} is closer to L_{orig}, and hence is of higher accuracy.

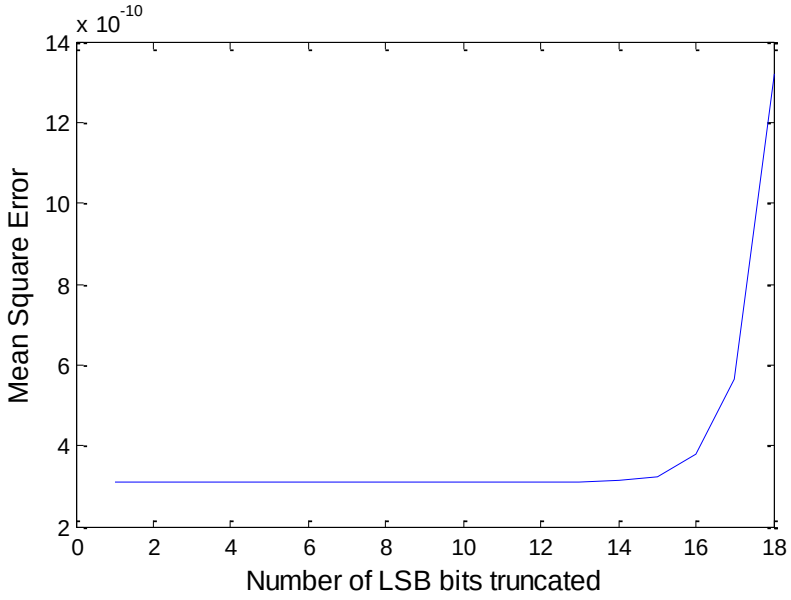


Figure 4-30: MSE for Gaussian filter output with different number of LSBs truncated.

Figure 4-30 shows that MSE increases with the number of LSBs truncated and remains constant until more than 12 bits are truncated. The MSE provides a quantitative measure of how much the truncation operation affects the scale space obtained, but provides no information about how much the detection result is affected as a function of the number of LSBs truncated. With this objective in mind, a more practical way is employed to check the effect of data truncation to see how the detection performance varies with the number of LSBs truncated at the output of Gaussian filter. The experimental settings are given in Table 4-6.

Table 4-6: Experimental settings for performance evaluation of data truncation on Gaussian smoothed pixels.

Settings	**Value**
Maximum Gaussian kernel size	31
Word length of Gaussian kernel coefficients	14 bits

Figure 4-31 shows that the detection performance remains unchanged until 18 LSBs are truncated, and the detection performance drops significantly when more than 22 bits are truncated. Therefore, 16 LSBs are truncated at the output, and each smoothed pixel is represented by 20 bits without loss of detection accuracy.

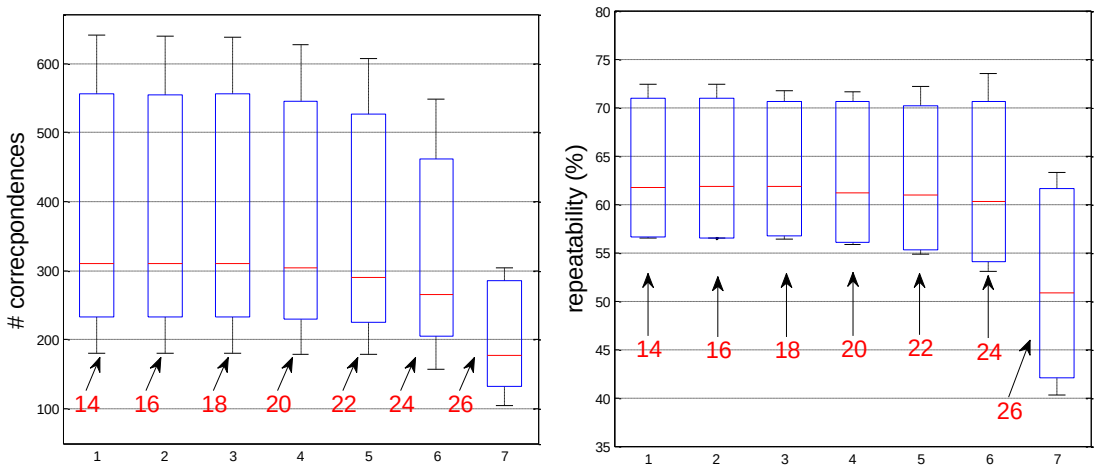


Figure 4-31: Detection performances as a function of the word length of Gaussian filtered pixels.

4.3.4 Error of Keypoint Detection with Stability Checking

The error of keypoint detection with stability checking refers to the comparison "Step 2" in Figure 4-24. This section presents the detection robustness as a result of the changes in the following aspects: (1) the word length of DoG (2) the number of iteration cycles for the location refinement process (3) the approximation in low contrast removal.

a. Truncation Error

The theoretical maximum word length of the DoG value is 21 bits, including a sign bit. Actually, the word length can be lowered by performing truncation on LSBs on

DoG values. Reducing the word length brings a reduction in the memory requirement for buffering DoG values.

Table 4-9 gives the settings for the experiments conducted in this section.

Table 4-7: Experimental settings for performance evaluation of DoG word length.

Settings	Value
Maximum Gaussian kernel size	31
Word length of Gaussian kernel coefficients	14 bits
LSBs truncation on Gaussian filtered images	16 bits

Figure 4-32 shows the detection results as a function of LSBs truncation that is performed on the DoG values prior to local extrema detection. The detection performance is virtually the same when less than 8 bits are truncated.

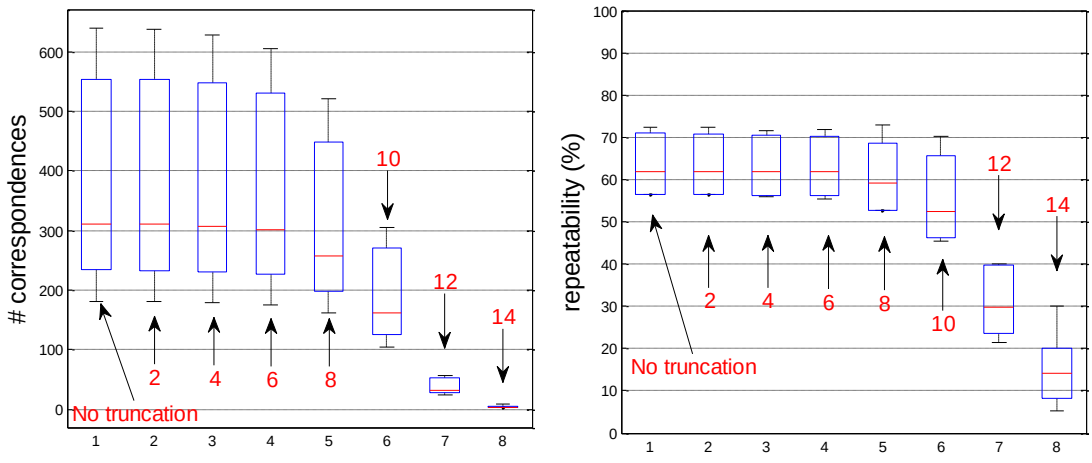


Figure 4-32: Correspondences and repeatability as a function of the number of LSBs truncated from DoG values.

Table 4-8 lists the block RAM usage for buffering DoG values as a function of the number of LSBs truncated for VGA sized images. With 6 bits truncated, the memory usage is reduced without significant degradation on the robustness of the feature detection. The corresponding word length of the DoG values is 15 bits.

Table 4-8: Block RAM consumption as a function of the number of LSBs truncated from DoG values.

Number of LSBs truncated	RAMB36E1	RAMB18E1
0	7	1
2	6	1
4	6	0
6	4	3
8	3	3
10	4	0

b. Location Refinement Process

Each local extremum $\mathbf{x} = (x, y)^T$ detected from the DoG scale space is passed to the location refinement process, where interpolation is performed on the location of the extremum. Output of each interpolation process is the offsets between the interpolated location and that of the origin. If the offset is larger than 0.5 in any dimension, the extremum is shifted to a new location $\mathbf{x}' = (x', y')^T$ by adding the offsets $(\Delta x, \Delta y)$ to the origin and repeats the interpolation process until the maximum number of iterations is hit.

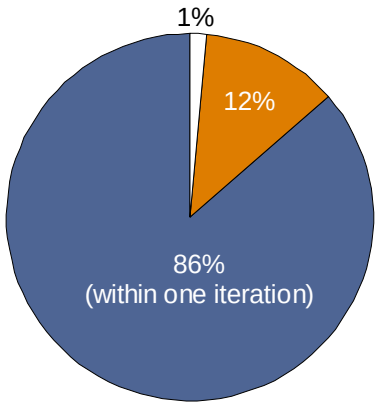
Five iterations are used in the reference model. However, it has been tested that the maximum number of iterations can be reduced from five to one, which significantly reduces the processing time at the cost of a little loss in localisation accuracy.

Table 4-9 lists the settings for the experiments conducted in this section.

Table 4-9: Experimental settings.

Settings	Value
Maximum Gaussian kernel size	31
Word length of Gaussian kernel coefficients	14 bits
LSBs truncation on Gaussian filtered images	16 bits
LSBs truncation on DoG values	3 bits

It is concluded from the experiments that the location refinement processes for around 86% of total extrema are completed within only one iteration cycle, indicating that the detected points are the local extrema under sub-pixel accuracy and are not refined to an adjacent location. Another 12% are fixed within two iterations, indicating that the extrema are shifted to an adjacent location. Only around 1% of the extrema requires more than two iteration cycles, as shown in Figure 4-33.



.

Figure 4-33: The location refinement process of around 86% of the total detected extrema is completed within only one iteration cycle, and 12% is finished within two iterations. Only around 1% takes more than two iterations.

Figure 4-34 shows the overall probability distribution of Δx and Δy. The majority of offsets falls in the interval [-0.5, 0.5], indicating that most of the detected points are

the local extrema under sub-pixel accuracy and are not refined to an adjacent location in any direction. Because the descriptors are computed from pre-defined scales instead of the closest scales (Section 4.2.3a), Δs is not analysed in this section.

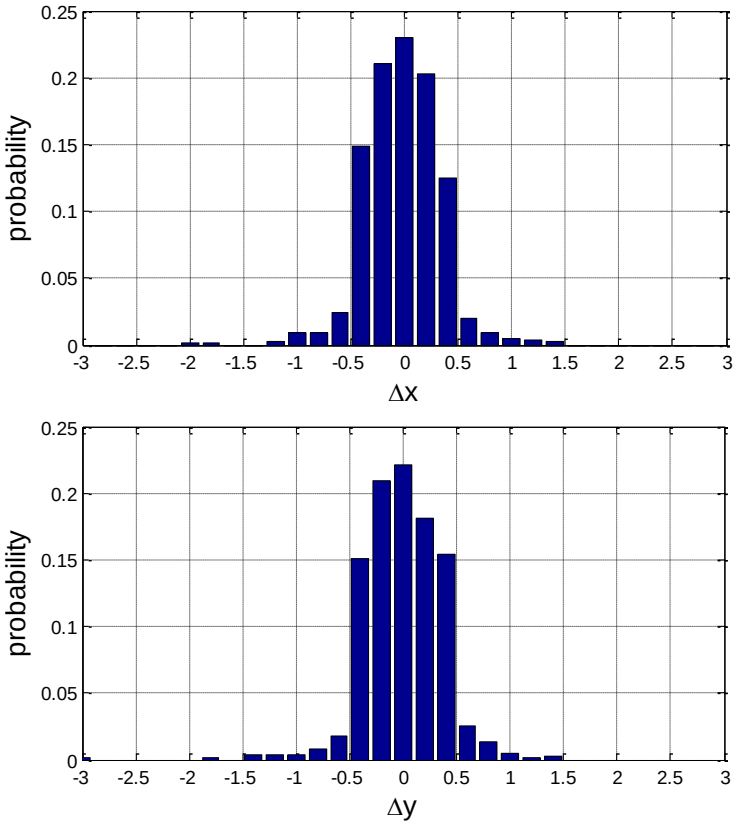
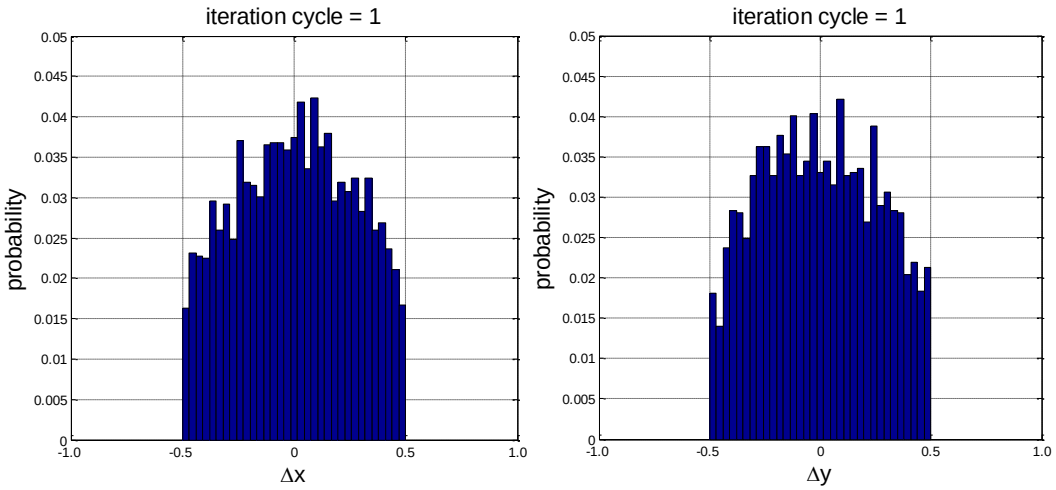
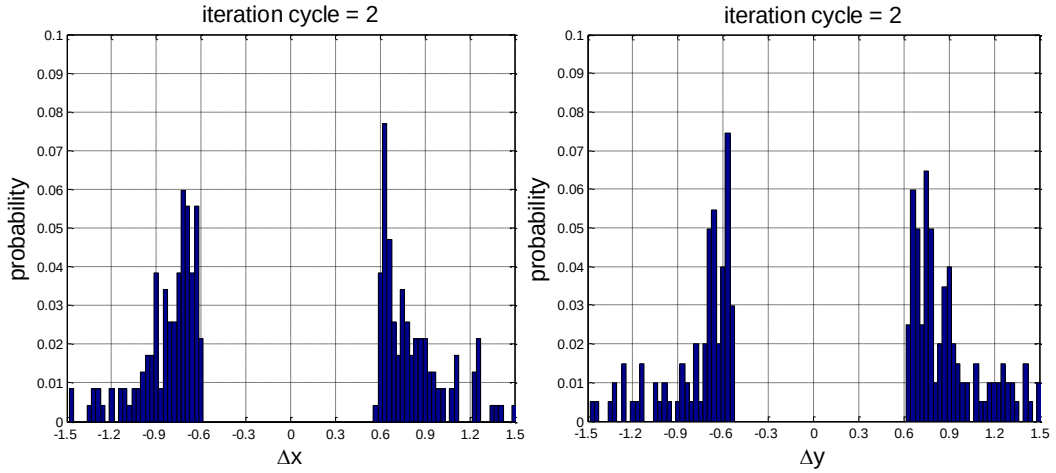


Figure 4-34: Overall probability distribution of Δx and Δy.

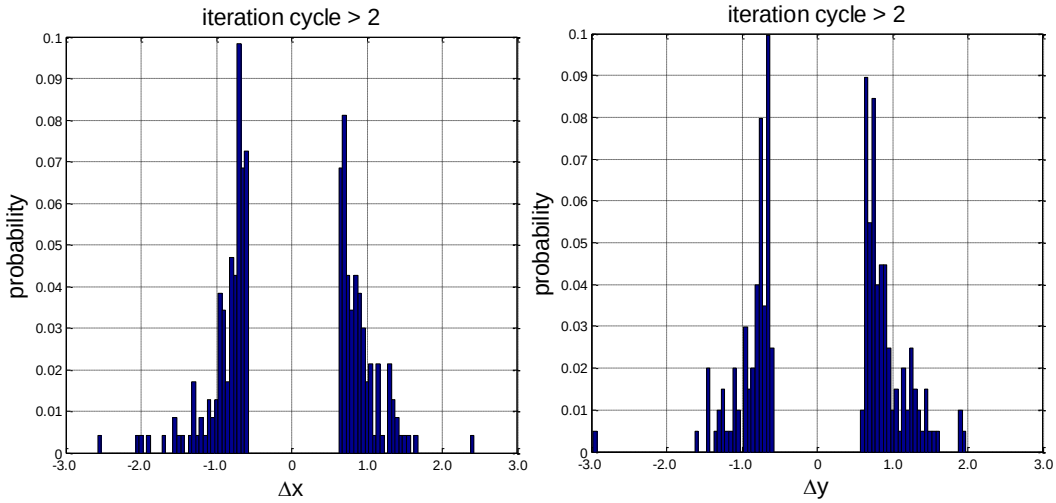
The effect of changing the number of iterations is investigated by looking at the probability distribution of offsets. Figure 4-35(a) and (b) show the distribution of Δx and Δy for refinement processes that are completed with one and two iteration cycles, respectively. Figure 4-35(c) shows the probability of Δx and Δy for refinement processes requiring more than two iteration cycles, which shows that only a few pixels are refined to a location that is more than one pixel away from the originally detected location. Therefore, the number of iteration cycles is limited to one, and the detected extrema is shifted to an adjacent pixel directly without further refinement if either Δx or Δy is beyond 0.5, with which approximately 99% of the original accuracy is kept.



(a) Probability of Δx and Δy for refinement process with one iteration cycle.



(b) Probability of Δx and Δy for refinement process with two iteration cycles.



(c) Probability of Δx and Δy for refinement process with more than two iteration cycles.

Figure 4-35: Probability distribution of Δx and Δy.

Although the processing time can be improved by exploring the parallel processing property within refinement process, successive iteration cycles for the same candidate keypoint have to be implemented in series, and hence the processing time is directly proportional to the number of iterations. As a result, by performing only one iteration cycle, the processing time is significantly reduced while keeping a high level of accuracy for feature detection.

Another advantage of reducing the maximum number of iteration cycles to one is that repeatedly computing the offsets $\hat{\mathbf{x}}$ can be avoided. The location of the extrema $\hat{\mathbf{x}}$ relative to the origin $\mathbf{x} = (x, y, s)^T$ is given below.

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}} \tag{4.6}$$

where

$$\frac{\partial D}{\partial \mathbf{x}} = \begin{bmatrix} D_x \\ D_y \\ D_s \end{bmatrix}$$

$$\frac{\partial^2 D}{\partial \mathbf{x}^2} = \begin{bmatrix} D_{xx} & D_{xy} & D_{xs} \\ D_{xy} & D_{yy} & D_{ys} \\ D_{xs} & D_{ys} & D_{ss} \end{bmatrix}$$

By substituting (4.7) into (4.6), it gives (4.8).

$$\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} = \frac{1}{\left|\frac{\partial^2 D}{\partial \mathbf{x}^2}\right|} \times \begin{bmatrix} \begin{vmatrix} D_{yy} & D_{ys} \\ D_{ys} & D_{ss} \end{vmatrix} & \begin{vmatrix} D_{xs} & D_{xy} \\ D_{ss} & D_{ys} \end{vmatrix} & \begin{vmatrix} D_{xy} & D_{xs} \\ D_{yy} & D_{ys} \end{vmatrix} \\ \begin{vmatrix} D_{ys} & D_{xy} \\ D_{ss} & D_{xs} \end{vmatrix} & \begin{vmatrix} D_{xx} & D_{xs} \\ D_{xs} & D_{ss} \end{vmatrix} & \begin{vmatrix} D_{xs} & D_{xx} \\ D_{ys} & D_{xy} \end{vmatrix} \\ \begin{vmatrix} D_{xy} & D_{yy} \\ D_{xs} & D_{ys} \end{vmatrix} & \begin{vmatrix} D_{xy} & D_{xx} \\ D_{ys} & D_{xs} \end{vmatrix} & \begin{vmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{vmatrix} \end{bmatrix} \tag{4.7}$$

$$\hat{\mathbf{x}} = \frac{1}{\left|\frac{\partial^2 D}{\partial \mathbf{x}^2}\right|} \times \begin{bmatrix} \begin{vmatrix} D_{yy} & D_{ys} \\ D_{ys} & D_{ss} \end{vmatrix} & \begin{vmatrix} D_{xs} & D_{xy} \\ D_{ss} & D_{ys} \end{vmatrix} & \begin{vmatrix} D_{xy} & D_{xs} \\ D_{yy} & D_{ys} \end{vmatrix} \\ \begin{vmatrix} D_{ys} & D_{xy} \\ D_{ss} & D_{xs} \end{vmatrix} & \begin{vmatrix} D_{xx} & D_{xs} \\ D_{xs} & D_{ss} \end{vmatrix} & \begin{vmatrix} D_{xs} & D_{xx} \\ D_{ys} & D_{xy} \end{vmatrix} \\ \begin{vmatrix} D_{xy} & D_{yy} \\ D_{xs} & D_{ys} \end{vmatrix} & \begin{vmatrix} D_{xy} & D_{xx} \\ D_{ys} & D_{xs} \end{vmatrix} & \begin{vmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{vmatrix} \end{bmatrix} \times \begin{bmatrix} -D_x \\ -D_y \\ -D_s \end{bmatrix} \tag{4.8}$$

The location refinement process is complicated in that it involves matrix inversion for computing $\hat{\mathbf{x}}$, which is expensive to be implemented on FPGA devices. With the

maximum number of iterations reduced to one, there is no need to compute the exact offset $\hat{\mathbf{x}}$ from the origin. The refined location can be determined by checking the relationship between the offset $\hat{\mathbf{x}}$ from the origin and 0.5. If $\mathrm{abs}(\hat{\mathbf{x}})$ is larger than 0.5 in any dimension, the extremum is closer to a neighbouring pixel. Therefore, matrix inversion can be avoided by rearranging (4.8) into (4.9), with which the division operation involved in matrix inversion is replaced by a comparator for each dimension, and hence the implementation efficiency is improved.

$$abs\left\{\begin{bmatrix}\begin{vmatrix}D_{yy} & D_{ys}\\ D_{ys} & D_{ss}\end{vmatrix} & \begin{vmatrix}D_{xs} & D_{xy}\\ D_{ss} & D_{ys}\end{vmatrix} & \begin{vmatrix}D_{xy} & D_{xs}\\ D_{yy} & D_{ys}\end{vmatrix}\\ \begin{vmatrix}D_{ys} & D_{xy}\\ D_{ss} & D_{xs}\end{vmatrix} & \begin{vmatrix}D_{xx} & D_{xs}\\ D_{xs} & D_{ss}\end{vmatrix} & \begin{vmatrix}D_{xs} & D_{xx}\\ D_{ys} & D_{xy}\end{vmatrix}\\ \begin{vmatrix}D_{xy} & D_{yy}\\ D_{xs} & D_{ys}\end{vmatrix} & \begin{vmatrix}D_{xy} & D_{xx}\\ D_{ys} & D_{xs}\end{vmatrix} & \begin{vmatrix}D_{xx} & D_{xy}\\ D_{xy} & D_{yy}\end{vmatrix}\end{bmatrix}\times\begin{bmatrix}-D_x\\ -D_y\\ -D_s\end{bmatrix}\right\} > abs\left\{0.5\times\left|\frac{\partial^2 D}{\partial\mathbf{x}^2}\right|\right\} \tag{4.9}$$

Table 4-10: Derivatives of *D*.

Derivatives	**Computation**
D_x	$[D(x+1,y,s)-D(x-1,y,s)]/2$
D_y	$[D(x,y+1,s)-D(x,y+1,s)]/2$
D_s	$[D(x,y,s+1)-D(x,y,s-1)]/2$
D_{xx}	$D(x+1,y,s)+D(x-1,y,s)-2D(x,y,s)$
D_{yy}	$D(x,y+1,s)+D(x,y-1,s)-2D(x,y,s)$
D_{ss}	$D(x,y,s+1)+D(x,y,s+1)-2D(x,y,s)$
D_{xy}	$[(x+1,y+1,s)+D(x-1,y-1,s)-D(x-1,y+1,s)-D(x+1,y-1,s)]/4$
D_{xs}	$[(x+1,y,s+1)+D(x-1,y,s-1)-D(x-1,y,s+1)-D(x+1,y,s-1)]/4$
D_{ys}	$[(x,y+1,s+1)+D(x,y-1,s-1)-D(x,y-1,s+1)-D(x,y+1,s-1)]/4$

The Hessian and derivative of D can be approximated by using differences of neighbouring DoG values, with which $\hat{\mathbf{x}}$ can be resolved with minimal cost, as shown in Table 4-10. Therefore, both the processing time and computational complexity are reduced as a result of limiting the number of iteration cycles for location refinement process to one instead of five.

In short, by reducing the number of iterations to one, the average processing time is reduced below half the original one. With the matrix inversion avoided by replacing division operation with comparison operator, the hardware efficiency is improved as a result of reduction in processing time and computational complexity.

c. Low Contrast Extrema Removal

In the standard SIFT algorithm, extrema with a value of $|D(\hat{\mathbf{x}})|$ less than 0.03 will be discarded as they are unstable with low contrast, assuming image pixel values are in range [0, 1]. The contrast at the extremum is defined below.

$$D(\hat{\mathbf{x}}) = D + \Delta D \tag{4.10}$$

with

$$\Delta D = \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\ \hat{\mathbf{x}}$$

where $\hat{\mathbf{x}}$ has been given in Equation (4.6), which is the offset from the refined location to the origin.

Because the computational complexity of the location refinement process has been reduced by replacing the division operation by a comparator, the exact value of $\hat{\mathbf{x}}$ is not calculated for hardware efficiency. To further reduce the computational complexity, Equation (4.10) is approximated by $D(\hat{\mathbf{x}}) = D$. Figure 4-36 shows the probability distribution of the ratio of $|\Delta D|$ to $|D(\hat{\mathbf{x}})|$, which is below 5% in most cases.

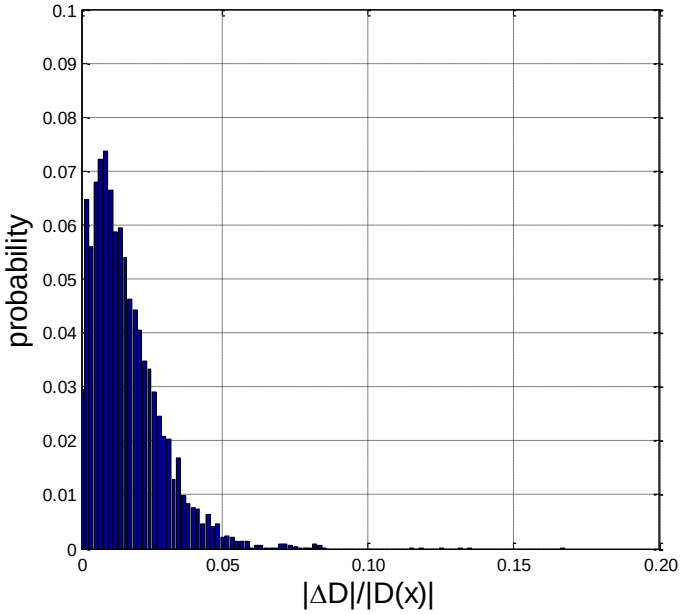
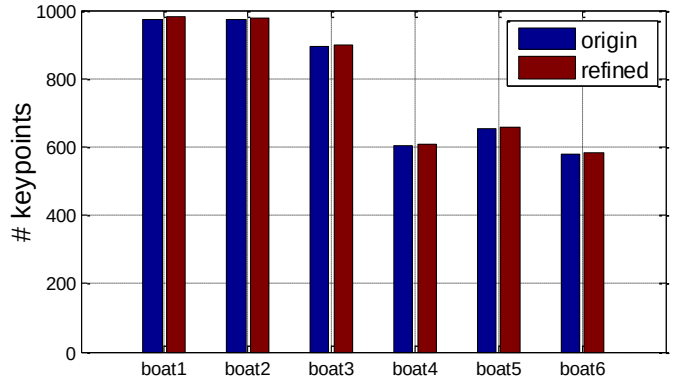


Figure 4-36: Probability of the ratio between $|\Delta D|$ and $|D(\hat{\mathbf{x}})|$.

The contrast at the refined location is always larger than that at the origin, which can be expressed as $|D(\hat{\mathbf{x}})| > |D|$. Therefore, by eliminating extrema with $|D|$ below the pre-defined threshold instead of using $D(\hat{\mathbf{x}})$ for low contrast removal, extrema with contrast $|D(\hat{\mathbf{x}})|$ slightly greater than the pre-defined threshold may be eliminated if $|D|$ is less than 0.03, which can be expressed as $|D(\hat{\mathbf{x}})| >0.03> |D|$. Because the difference between $|D|$ and $|D(\hat{\mathbf{x}})|$ is rather small, as shown in Figure 4-36, the number of keypoints is not reduced significantly as a result of approximating $|D(\hat{\mathbf{x}})|$ with $|D|$, as shown in Figure 4-37.



..

Figure 4-37: Comparison of detection results for low contrast removal.

By discarding low contrast points based on the original extrema, the low contrast removal process is no longer dependent on the location refinement process, and hence these two processes can be processed in parallel. As a result, the throughput of stability checking process is improved at the cost of a slight degradation in detection performance.

4.3.5 Overall Comparison for Feature Detection

The overall detection performance is compared as a result of the above mentioned approximations, which corresponds to the comparison "Step 3" shown in Figure 4-24. Table 4-11 lists the experimental settings for Lowe's software model and the FPGA design.

Table 4-11: Experimental settings for feature detection.

Settings	Lowe's Model	FPGA Design
Maximum Gaussian kernel size	43	31
Word length of Gaussian kernel coefficients	Floating-point	14 bits
LSBs truncation on Gaussian filtered images	0	16 bits
Word length of DoG values	Floating-point	21 bits
Maximum number of iterations for localisation refinement of keypoints	5	1
Approximation on low contrast removal	No	Yes

Figure 4-38 shows the comparison of correspondence and repeatability between the software model and the FPGA design. The FPGA design keeps the detection robustness at a similar level to that of the software model. The FPGA design provides a larger number of correspondences and a higher repeatability mainly because of the use of smaller Gaussian kernel (k_G=31) that causes slight over-detection.

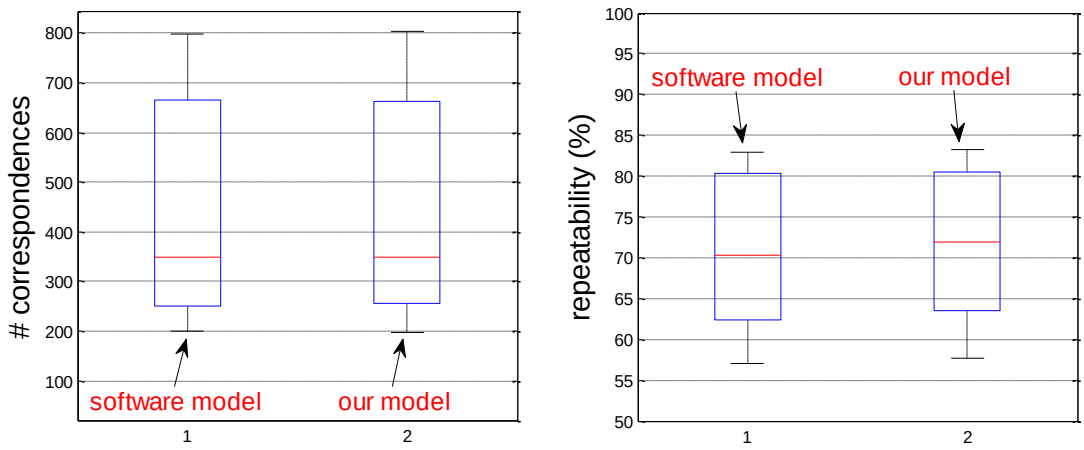


Figure 4-38: Comparison of detection and matching performance between software model and the FPGA design.

4.3.6 Simulation Scheme for Descriptor Generation

As shown in Figure 4-39, three comparisons are performed to evaluate the performance of the hardware design for the descriptor generation module. The Normalised Descriptor Vector Generation block consists of three units: 36-bin histogram generation, linear interpolation and descriptor computation.

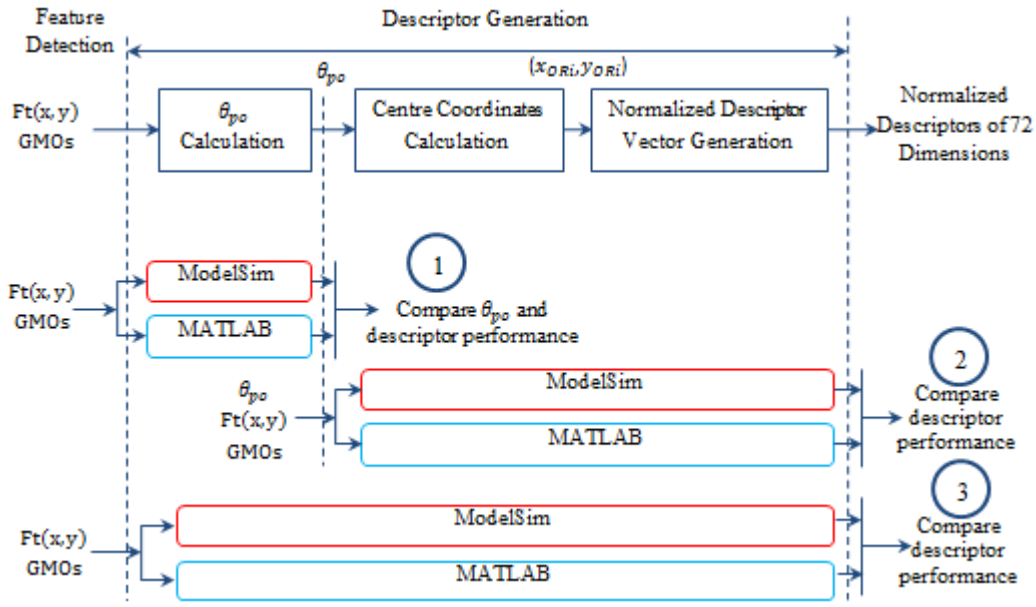


Figure 4-39: Three comparisons to evaluate the processing accuracy of descriptor generation.

4.3.7 Precision of Principal Orientation Calculation

In this section, experiments are conducted to see how the principal orientation is affected by the precision of GMOs, which refers to the comparison "Step 1" as shown in Figure 4-39. There are two sources of errors: 1) the error introduced by the approximation based computation method 2) the quantisation error caused by using fixed-point accuracy with limited word length. Experiments are conducted to check the effect of GMOs on descriptor generation, and further the matching performance.

a. Approximation Error

Initially, GMO computation involves complicated operations, such as division, square root computation and arctan function. Without any optimisations, it is considerably expensive for these operations to be realised on hardware devices. Therefore, the approximation based computation method is proposed to reduce the computational complexity. Calculation errors are introduced into the system by using the approximation based method for GMO computation, including the shift register (SRT) based square root calculation for gradient magnitude, and the LUT-based gradient orientation computation.

SRT-based Square Root Calculation

The relative error of the SRT-based square root calculator is given in Figure 4-40, which is generated by taking the ratio of the result from the SRT-based square root computer to that of the double precision floating-point model. In general, the relative error decreases with the increase of the radicand and falls below 1% when the radicand is around 30,000 that corresponds to 15 bits in binary. For better precision, more bits must be used in calculation, which brings up the trade-off between the precision and the processing time. Detailed introduction to the SRT-based square root computer will be given in Chapter 5.

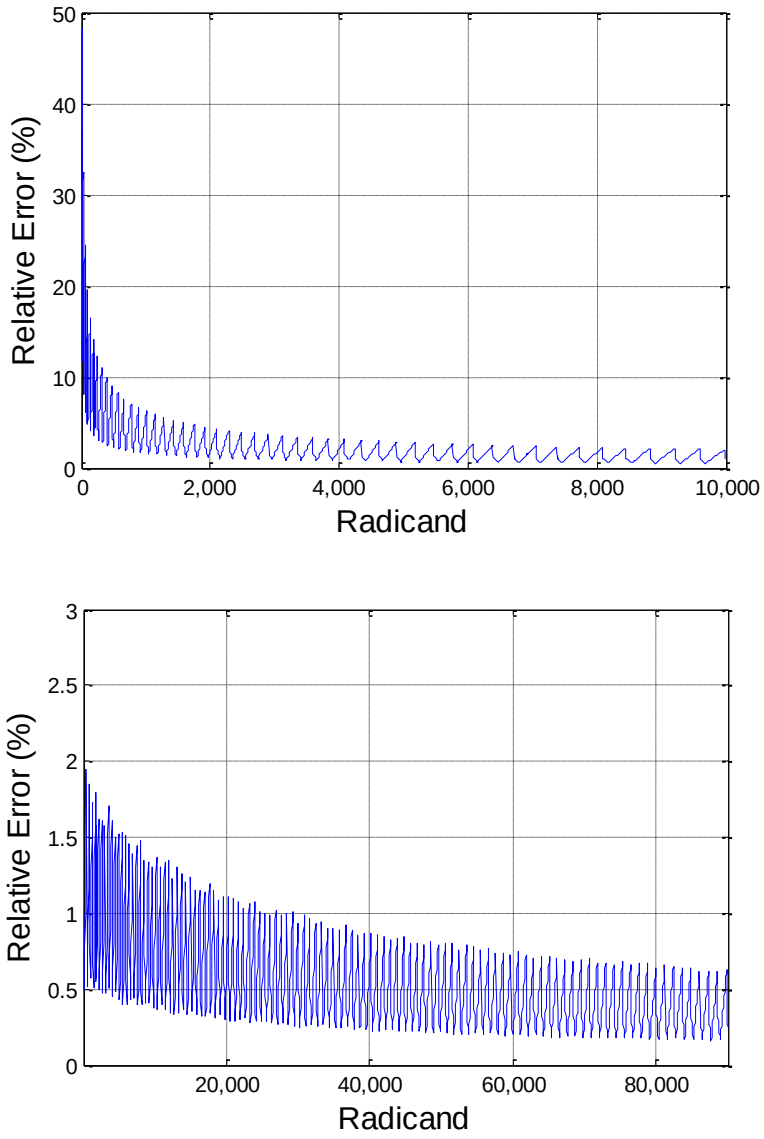


Figure 4-40: Relative error of the SRT-based square root calculation.

Because each Gaussian smoothed pixel is represented by 20 bits and the word length of the radicand for gradient magnitude calculation is double that of the smoothed pixel, the relative error caused by using the SRT-based square root calculation is actually small enough to be safely ignored. Figure 4-41 shows the error for the gradient magnitude calculated by using the SRT-based square root calculator relative to the floating-point calculation for the reference image from the boat sequence. The x-coordinate is the row index to the image, and the y-coordinate shows the relative error from the corresponding row of image as a result of using SRT-based square

root calculation. The relative error of the entire image is below 0.01% and is small enough to be safely ignored.

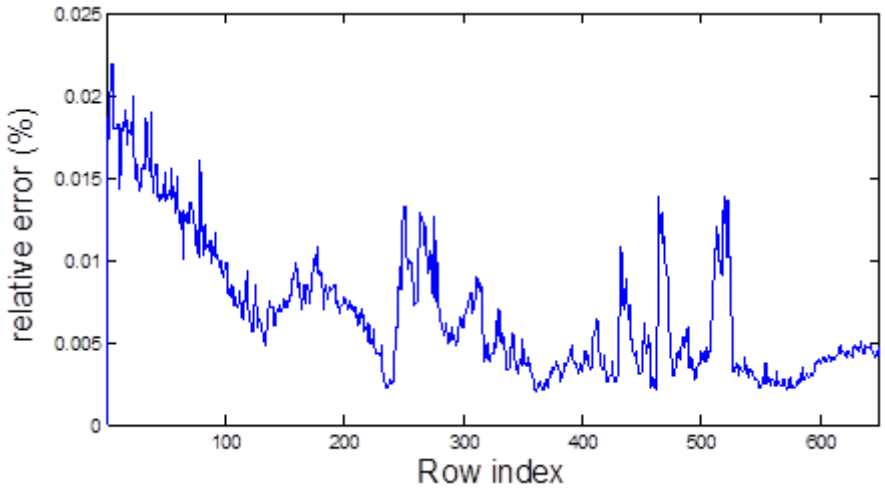


Figure 4-41: Relative error produced by comparing the gradient magnitudes calculated using the SRT-based square root solution with those produced by the MATLAB model.

LUT-based Orientation Calculation

The gradient orientation is calculated using Equation (2.13).

$$\theta(x,y) = \tan^{-1}(\frac{G_y}{G_x}) \tag{4.11}$$

The gradient orientation $\theta(x,y)$ is inefficient to be computed on hardware since it includes division operation and arctan computation that are hardware-expensive and time-consuming. Inspired by the fact that the orientation is quantised to 36 directions with each representing 10^o, there is no need to compute the exact gradient orientation of each pixel. Instead, the quantised orientation that a pixel belongs to is computed directly by using the LUT-based strategy, which is fast to compute while keeping the initial precision.

The quantised orientation θ_t in range [0, 8] is first considered. Taking advantage of the monotonically increasing property of tan function in range [0, 90^o], θ_t can be decided by comparing its tangent value with predefined thresholds:

$$\tan(\theta_i) \leq \tan(10 \cdot \theta_t) \leq \tan(\theta_{i+1}) \tag{4.12}$$

where

$$\theta_i = 10i, \qquad i \in [0, 8]$$

(4.12) can be arranged into (4.13) by substituting Equation (2.13) into (4.12).

$$\tan(\theta_i) \leq \frac{|G_y|}{|G_x|} < \tan(\theta_{i+1}) \tag{4.13}$$

To avoid division operation, (4.13) is further arranged into (4.14).

$$|G_x| \cdot \tan(\theta_i) \leq |G_y| < |G_x| \cdot \tan(\theta_{i+1}) \tag{4.14}$$

Therefore, the quantised orientation θ_t can be easily identified by comparing $|G_y|$ with pre-defined thresholds. If the relationship shown in (4.14) is satisfied, θ_t is set to i. Pixel orientation $\theta(x, y)$ can be easily identified by checking the sign of G_x and G_y, and hence the quantised orientation that a pixel belongs to can be identified by simple multiplication and comparison operations.

$$\theta(x, y) = \begin{cases} \theta_t, & G_x \geq 0, G_y \geq 0 \\ 17 - \theta_t, & G_x < 0, G_y \geq 0 \\ 18 + \theta_t, & G_x < 0, G_y < 0 \\ 35 - \theta_t, & G_x \geq 0, G_y < 0 \end{cases} \tag{4.15}$$

The error in principal orientation (θ_{po}) is checked as a result of the LUT-based calculation.

Figure 4-42 shows the comparison results between the LUT-based method and the floating-point model using atan function. In the LUT-based method, the orientations are quantised to integers in range 0 to 35 with each representing 10^o. When compared with the results from the floating-point model, the error ($\Delta\theta(\mathrm{x}, \mathrm{y})$) in orientation is

always less than 0.5. With the orientations from the floating-point model also quantised to integers in range 0 to 35, the difference is eliminated, as shown in the bottom image of Figure 4-42. Therefore, the LUT-based method is able to provide quantised orientations that are of the same accuracy with that of the floating-point model.

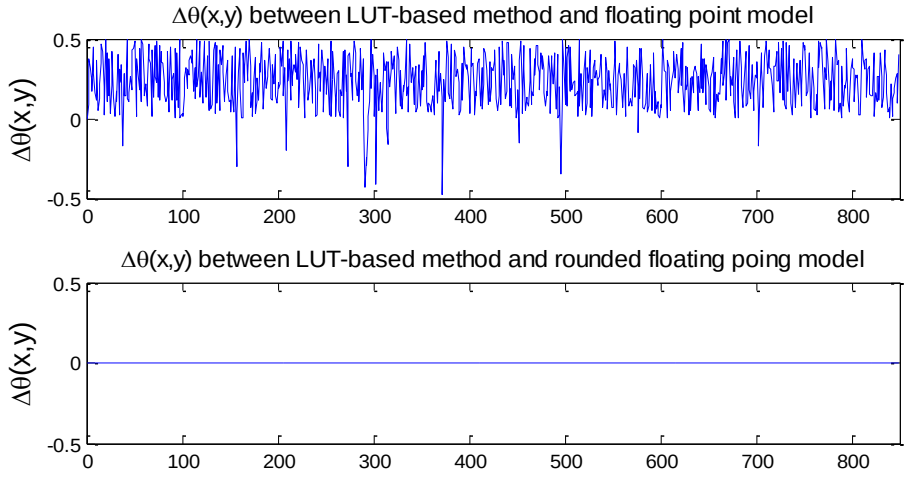
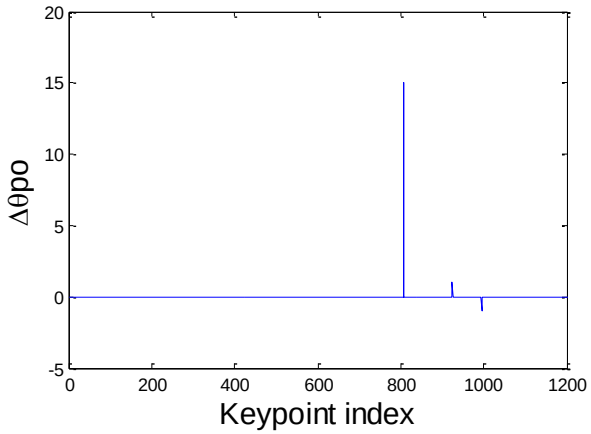


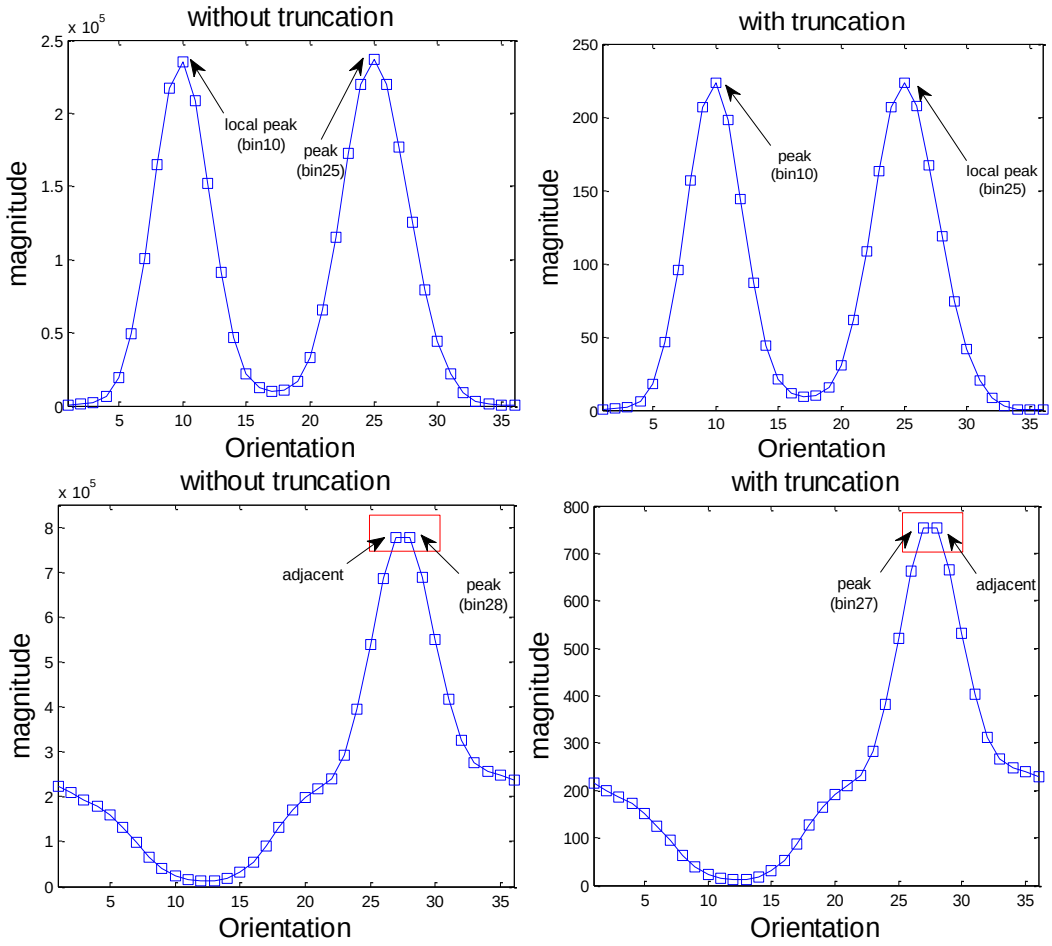
Figure 4-42: Comparisons of the pixel orientations calculated by using the LUT based method and that from the MATLAB model using atan function.

b. Fixed-point Error

The theoretical maximum word length of the gradient magnitude is 21 bits with the input filtered pixels represented by 20 bits. In practice, truncation is performed on the LSBs of the filtered pixels before the gradient magnitude calculation. This is to reduce the time requirement of the SRT-based square root calculator, which is proportional to the word length of the radicand. Besides, the resultant word length of gradient magnitude is reduced as well, which reduces the memory requirement for buffering GMOs as well as the throughput requirement of the DDR3 that is proportional to the word length of GMO. The final gradient magnitude is represented by 10 bits. It should be noticed that the input to the LUT-based orientation calculation is still 20 bits.



(a) error in principal orientation calculation for keypoints



(b) 36-bin histogram of the local region for principal orientation computation with multiple peaks of similar value

Figure 4-43: Comparison of the results generated with and without truncation performed on the LSBs of Gaussian filtered pixels.

The comparisons between the results generated with and without data truncation is shown in Figure 4-43, which shows the error in θ_{po} as a result of the data truncation. It can be seen from Figure 4-43(a) that $\Delta\theta_{po}$ is zero in most cases, which indicates that the same orientation is defined as θ_{po} whether the truncation is performed or not. There are several outliers where another orientation is defined as θ_{po}. The outlier occurs mainly due to the possibility that the 36-bin histogram of the local region has two bins of similar accumulated magnitude, as shown in Figure 4-43(b). This can be compensated by creating keypoints for any local peak that is within 80% of the highest peak of the 36-bin histogram, with which there might be multiple keypoints created at the same location but with different orientation. As a result, the orientation assignment is not significantly affected by reducing the word length of Gaussian filtered pixels.

With the gradient magnitude and orientation represented by 10 bits and 6 bits, respectively, each GMO can be represented by 16 bits. Therefore, four GMOs can be concatenated and sent to DDR3 as a single data to make full use of the data width (64 bits).

The previously presented results show the effect of the approximation based GMO computation and the reduced accuracy of gradient magnitude on the error in θ_{po} computation from a more theoretical perspective. Because the rotation invariance depends on the precision of θ_{po}, experiments are conducted to check the rotation invariance of descriptors as a result of the above mentioned errors. The recall versus 1-precision curve is used, which has become the golden standard for descriptor performance evaluation. It can be seen from Figure 4-44 that the descriptor performance remains virtually the same.

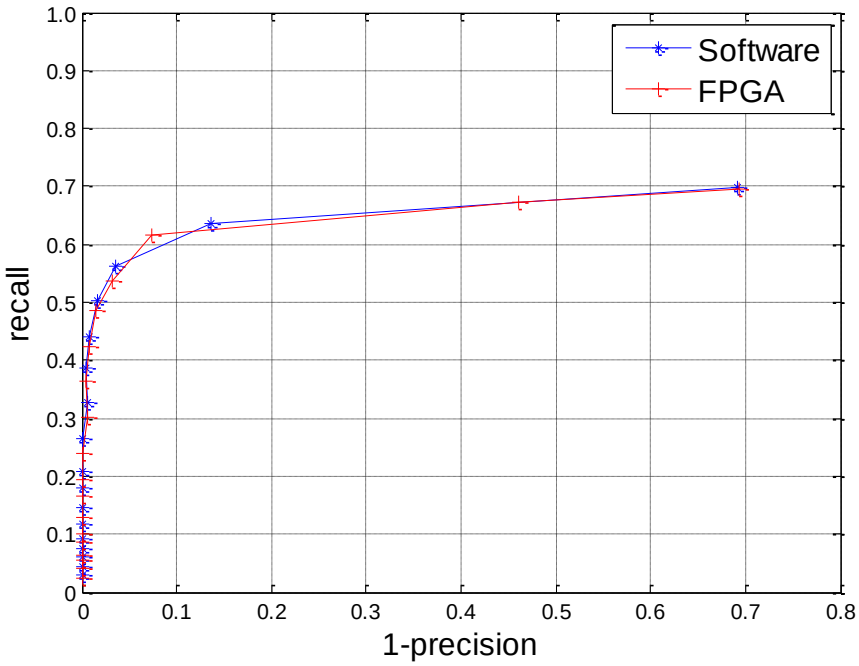
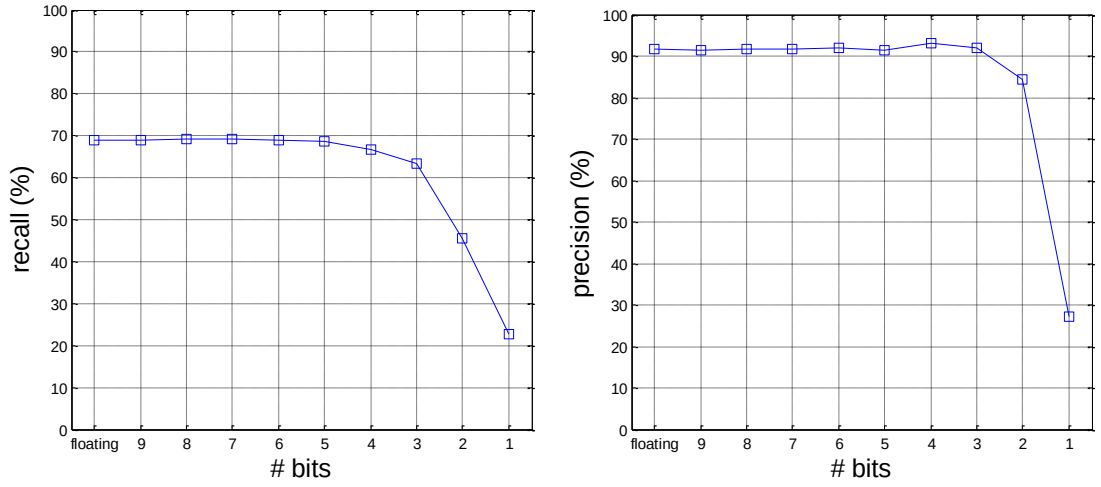


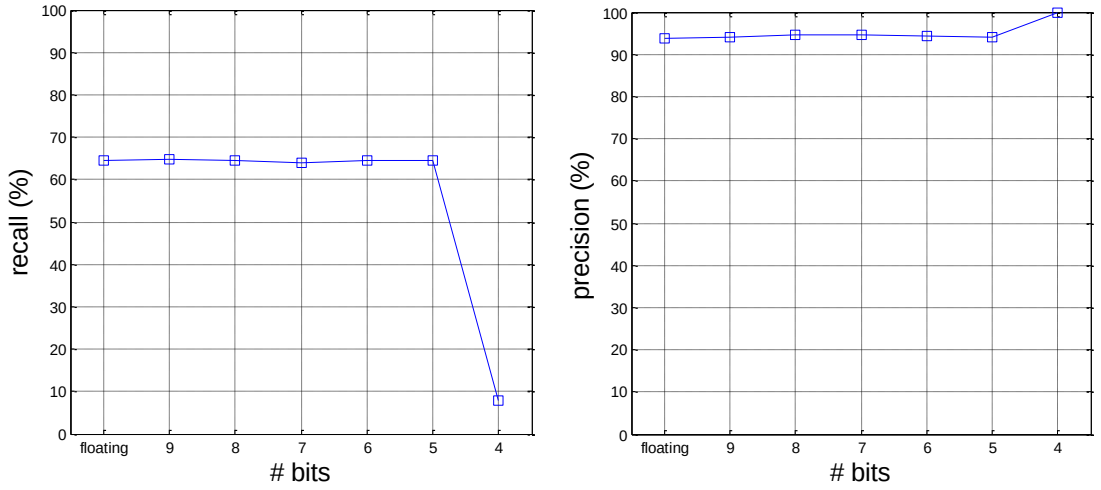
Figure 4-44: Comparison of recall versus 1-precision curve between software model and the FPGA design for "Step 1".

4.3.8 Quantisation Error of Feature Descriptor

The feature descriptor is quantised by reducing the number of bits representing each descriptor. The aim of the quantisation is to reduce the memory requirement for storing descriptors with minimal loss of matching reliability. To eliminate the effect of matching strategy on the results, experiments are conducted using both the distance ratio based method and the novel matching strategy. The results are given in Figure 4-45, showing the matching performance as a function of the word length of the final descriptors. For the SIFT-based method, the matching results remain virtually the same when representing descriptors by at least 3 bits. For the novel matching strategy presented in Chapter 3, the results are rather stable when the word length is at least 5 bits. Therefore, the suggested word length of the final descriptors is 3 bits and 5 bits for the SIFT-based method and the proposed strategy, respectively.



(a) Matching performance using the SIFT-based method.



(b) Matching performance using the proposed matching strategy.

Figure 4-45: Matching performance as a function of the word length of descriptors.

Figure 4-46 shows that the Block RAM usage is proportional to the word length of the normalised descriptors. Although the proposed matching strategy consumes more RAM for buffering descriptors than the SIFT-based matching, the computational complexity is reduced and resources are saved, such as DSP48E1. With a trade-off made between matching performance, processing complexity and memory usage, each descriptor is represented by 5 bits and the matching is carried out using the proposed matching strategy.

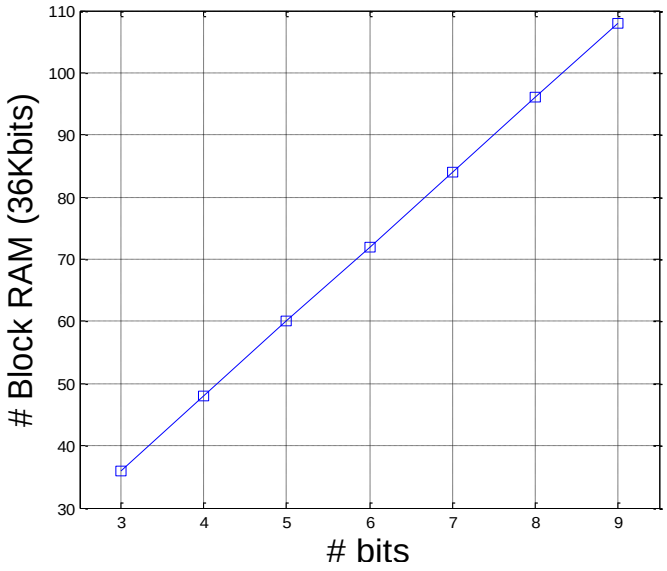


Figure 4-46: Block RAM (36 Kbits) consumption as a function of the word length of descriptors.

4.3.9 Overall Comparison for Descriptor Generation

Figure 4-47 shows the comparison of the recall versus 1-precision curve between the software model and the FPGA design, which corresponds to "Step 3" shown in Figure 4-39. The curve of the FPGA design is slightly below that of the software model as a result of the limited word length of Gaussian coefficients, the approximation based GMO computation, and the reduced word length of descriptors.

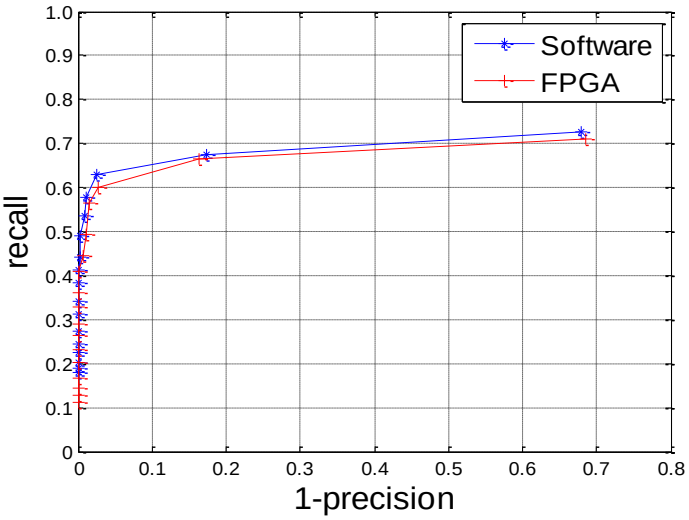


Figure 4-47: Comparison of recall versus 1-precision curve between software model and the FPGA design.

4.4 Summary

In this chapter, design parameters are studied for both keypoint detection and descriptor generation. The feature detection module is parameterised with two octaves of five Gaussian smoothed images each. Instead of computing the descriptors based on the closest scale of the keypoint, the pre-defined scale is used for descriptor generation, which reduces both the memory requirement and the processing time to a lower level at a cost of a little loss in matching performance.

In an FPGA-based implementation, there is always a trade-off between the processing accuracy and the hardware performance, such as resource usage and system throughput. The designer needs to balance the required performance against the implementation cost. With a trade-off made between throughput and accuracy, the size of largest Gaussian kernel is limited to k_G=31, which achieves a throughput of at least 60 fps with two pixels processed in parallel. Detailed description of the relationship between k_G and system throughput will be given in Chapter 5. The Gaussian filter process has been chosen to have input and output of 8 bits and 20 bits, respectively, with data truncation performed on both MSBs and LSBs to reduce computation cost. Error introduced by quantised Gaussian filter coefficients is reduced by representing the coefficients by 14 bits. Another example of the trade-off between accuracy and throughput is that the maximum number of iteration cycles for location refinement process is limited to one, which reduces the processing time at the cost of a little loss in performance. Besides, an approximation based method has been proposed for GMO computation. Since the time consumption of the SRT-based square root calculator is proportional to the word length of the radicand, the time requirement is reduced to half by truncating 10 LSBs of the Gaussian filtered pixels for gradient magnitude calculation at the expense of a slight degradation in matching performance. When quantised to integers in range 0 to 35 with each representing 10^o, the gradient orientation computed using the LUT-based method has the same accuracy with the floating-point model using atan function. An example of the trade-off between accuracy and resource usage is to use the fixed-point data format that consumes less hardware resource usage at the expense of a slight degradation in computation accuracy. Another example is to save the on-chip memory consumption by reducing the word length of the normalised descriptors. By representing each

descriptor with only 5 bits, the loss of matching accuracy is kept at a minimum level while keeping the matching accuracy at almost the original level.

Chapter 5 Processing Core of the Optimised SIFT Algorithm

5.1 Introduction

This chapter presents the detailed information on the processing core developed for the optimised SIFT algorithm. The core addresses the inefficient data acquisition and processing problem by offering a new pixel streaming method and a high level of parallel computation. Besides, novel memory access strategies are proposed for memory reduction.

The processing core proposed in this thesis is the first complete FPGA solution to the SIFT with all phases of the algorithm covered. By taking advantage of the parallel processing ability of FPGA, the design is able to process VGA video at least 60 fps, providing that there are no more than 2,200 keypoints per frame. The design is fully mapped to a Xilinx Virtex-6 FPGA device.

5.2 FPGA-Based SIFT Processing System

Taking advantage of the hardware resources and the high-level parallel processing capability provided by the FPGA technology, it is possible to embed the entire system into an FPGA device. The complete SIFT based image matching system is shown in Figure 5-1. The FPGA embedded system (in red) processes the images received from the camera and sends data to a host PC. The data could be the raw images received from the camera, or the matching results from the SIFT processing core. External memory (DDR3) is required as the internal memory in the FPGA device is insufficient for the system.

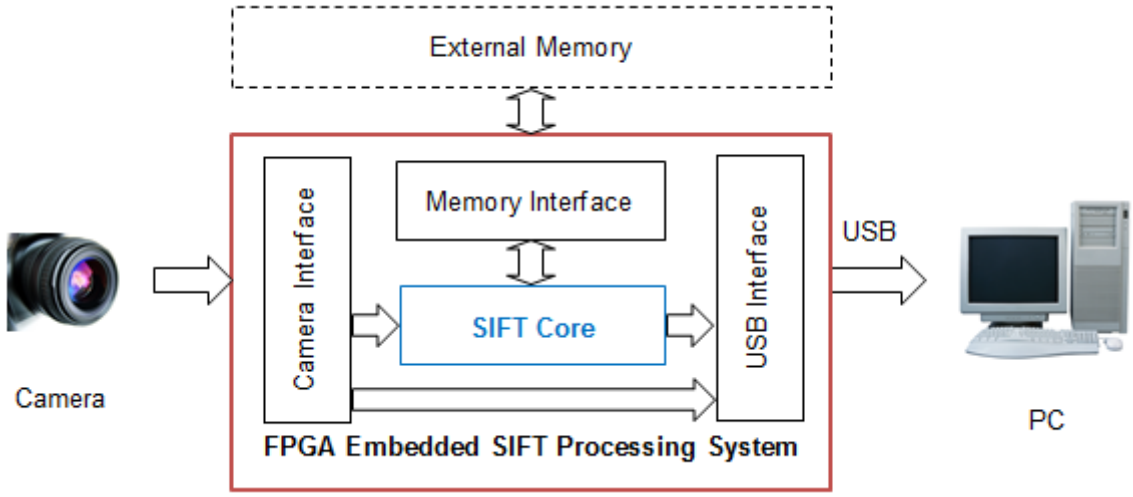


Figure 5-1: Block diagram of the SIFT based image matching system.

The FPGA embedded SIFT processing system mainly consists of the following blocks:

a. *Camera Interfacing Block*: This block interfaces with the commercial camera mounted on an Avnet Dual Image Sensor FMC Module to acquire the images which are streamed into an internal buffer.
b. *Memory Interfacing Block*: This block interfaces with the external memory on the Xilinx ML605 FPGA board to provide extra memory (DDR3) for intermediate processing results of the SIFT core.
c. *SIFT Processing Core*: The task of this core is to detect keypoints from the images acquired from the camera and further transfer the keypoints to distinctive descriptors that can be used for image matching. The output is the coordinates of the matched keypoints from images under consideration.
d. *USB Interfacing Block*: This block interfaces with USB controller chip to transfer both the raw images received from the camera and coordinates of matched keypoints to a host PC for display and further processing.

This chapter mainly focuses on an efficient SIFT processing core, and description of the FPGA based platform will be given in Chapter 6.

5.2.1 Field Programmable Gate Array Technology

The calculation requirement increases rapidly with image resolution, frame rate, and the number of keypoints to be processed in each frame. Furthermore, the amount of data transferred from the camera front-end to the USB back-end is extremely large if the image resolution and frame rate are required to be high. In order to obtain a high overall frame rate, an efficient processing method and data acquisition scheme needs to be applied. The data should be collected, processed and transferred concurrently without interruption. Using an FPGA device is an excellent solution to this requirement.

a. Overview of FPGA Device Block Structure

An FPGA is a semiconductor device that is based around a matrix of Configurable Logic Blocks (CLB) interconnected via programmable interconnects both horizontally and vertically. The device can be programmed to the desired function by users after manufacture, and hence the name "Field-Programmable". FPGAs have evolved far beyond the basic capabilities of its predecessors, such as DSP and ASIC. As shown in Figure 5-2, an FPGA device typically consists of an array of CLBs, interconnect routing, IO blocks (IOB), memory (BRAM), and digital clock management (DCM). The FPGA devices are generally programmed by using a Hardware Description Language (HDL), such as VHDL or Verilog.

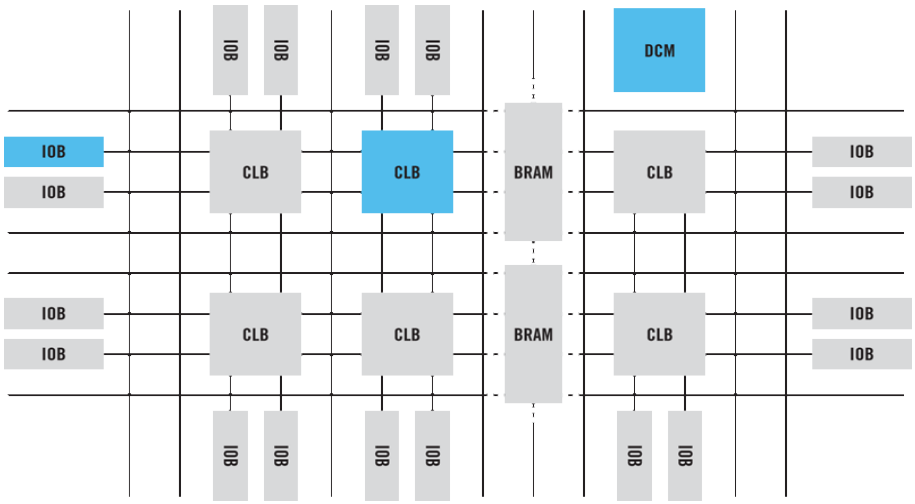


Figure 5-2: General architecture of an FPGA device [65].

b. Features of Xilinx FPGAs

Xilinx offers a broad range of FPGAs providing advance features, low-power, high-performance, and high capacity for any FPGA design. Below is an overview of Xilinx leading FPGA families, in terms of features of interest.

Table 5-1: Features of Xilinx FPGAs.

Features	**Artix™-7**	**Kintex™-7**	**Virtex-7**	**Spartan-6**	**Virtex-6**
Logic Cells	215,000	480,000	2,000,000	150,000	760,000
Block RAM	13 Mbits	34 Mbits	68 Mbits	4.8 Mbits	38 Mbits
DSP Slices	740	1,920	3,600	180	2,016
Memory Interface (DDR3)	1,066 Mbits/s	1,866 Mbits/s	1,866 Mbits/s	800 Mbits/s	1,066 Mbits/s
I/O Pins	500	500	1,200	576	1,200
I/O Voltage	1.2V, 1.35V, 1.5V, 1.8V, 2.5V, 3.3V			1.2V, 1.5V, 1.8V, 2.5V, 3.3V	1.2V, 1.5V, 1.8V, 2.5V

Table 5-1 shows that Virtex-7 family provides up to 2,000,000 logic cells and 68 Mbits Block RAM. These features, especially the number of Block RAMs available, are attractive to the hardware design for the SIFT processing core, which is demanding in hardware resources. However, with the DDR3 employed to deal with the large memory requirement, Virtex-6 FPGA meets the hardware resource requirement of the design for processing VGA images. The design can be migrated onto a Virtex-7 FPGA device for processing images of higher resolution, but this is beyond the scope of this project.

The Virtex-6 FPGA family is divided into three sub-families, each targeting on different features: Virtex-6 LXT FPGAs, Virtex-6 SXT FPGAs, and Virtex-6 HXT FPGAs. Every Virtex-6 FPGA has 156 to 1064 dual-port RAMs, each storing 36 Kbits. Each block RAM has two completely independent ports that share the stored

data [66]. Each port can be configured with one of the available widths, independent of the other port. This design used the Xilinx ML605 base board with the XC6VLX240T-FFG1156 FPGA as shown in Figure 5-3, which provides 241,152 logic cells, 37,680 slices, 768 DSP48E1 slices, and a maximum of 14,976 Kbits Block RAM.

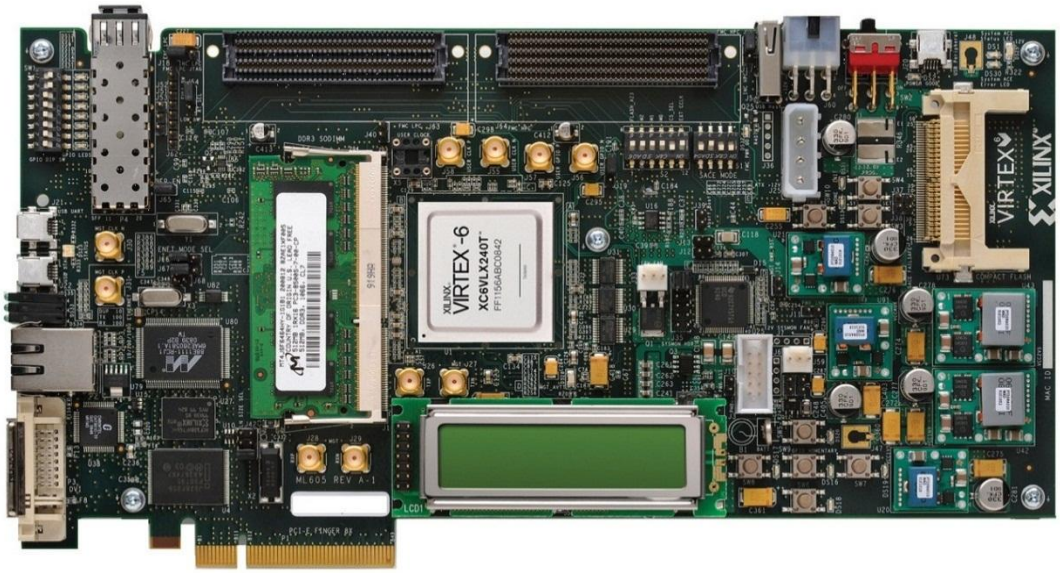


Figure 5-3: ML605 base board with the XC6VLX240T-FFG1156 FPGA.

5.2.2 Advantages of using FPGA

The main advantages of using FPGA in SIFT-based image processing system are:

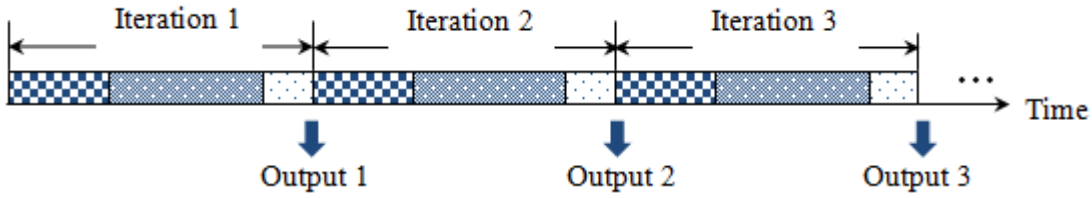
1. FPGAs have grown in capacity and performance, and have become a viable solution to computationally intensive tasks. SIFT is known for its promising performance. However, SIFT is of high computational complexity, making FPGA a viable choice. An example is the Gaussian scale space construction process, which requires a huge number of operations that makes it extremely difficult to achieve the real-time processing target when a serial computing device is used, such as PC or DSP.
2. The SIFT processing core can be programmed to perform concurrently and in a pipelined fashion. This feature takes advantage of the inherent parallel processing property of FPGA devices, with which the sub-modules of SIFT

algorithm can be implemented in parallel to achieve high throughput. For example, the Gaussian filters are applied to each pixel independently to generate the Gaussian pyramid. Therefore, these independent processes can be performed in parallel to reduce the total computation time. With the sub-modules of SIFT algorithm arranged into pipelined architecture, the throughput is further improved. For example, the SIFT feature detection part takes M clock cycles, and the SIFT descriptor generation takes N clock cycles to generate descriptors based on the output from the feature detection. The best case is $M = N$ so that the output of feature detection part can be continuously streamed into the descriptor generation. In this case, one set of descriptors are outputted after every M clock cycles, which is the maximum achievable performance of the SIFT processing core. Detailed description of the pipeline architecture will be given in Section 5.3.

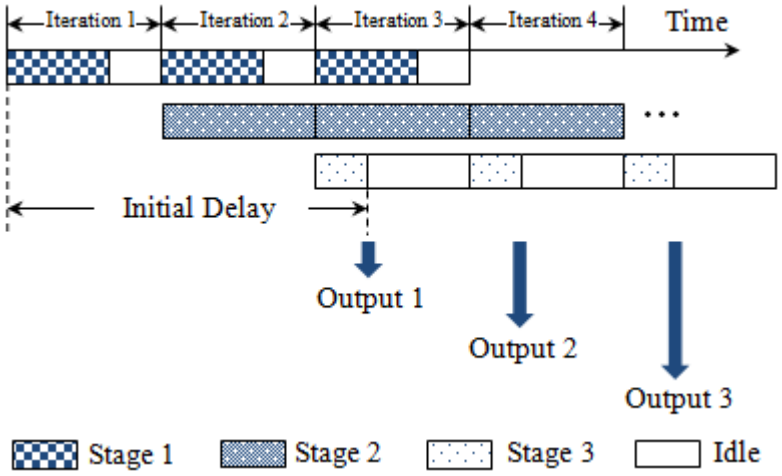
3. Xilinx support a wide range of embedded processing IP cores that works directly in a software tool called Xilinx Platform Studio (XPS). XPS is an integrated environment that contains a wide variety of embedded tools, IP cores and libraries to quickly create and develop an embedded system inside FPGA devices. The IP cores can be easily included in a design project to shorten the design cycle. For example, the Multi-Port Memory Controller (MPMC) provides fully parameterisable access to external memory, such as DDR3 on Xilinx Virtex-6 FPGA board. Moreover, there are many processor IP cores available for FPGAs that considerably extend the functionality of the system, such as MicroBlaze and PowerPC 440. Besides, a customised IP core can be integrated into a MicroBlaze based system, with which it is possible to build a highly compact and easy-to-access system on an FPGA device. Compared with the mask programmable ASIC technology, it is fast, convenient and flexible to develop an embedded system in an FPGA device.
4. FPGAs are available in a wide range of sizes with different features. An FPGA device can have more than one thousand I/O pins, which supports many standard I/O interfaces. Therefore, it is straightforward to interface to external devices off the board for functionality extension, such as the commercial camera and the USB controller board.

5.3 Hardware Architecture of the SIFT Processing Core

This section describes the SIFT processing core that is fully embedded in an FPGA device. The core is developed following the parameters considered and analysed in Chapter 4. The pipeline strategy is employed for high throughput, which is the most important technique used by reconfigurable systems.



(a) Non-Pipelined Architecture



(b) Pipelined Architecture

Figure 5-4: Block diagram of the non-pipelined and the pipelined architecture.

Figure 5-4 compares the non-pipelined and the pipelined architecture of a three-stage design. As shown in Figure 5-4(a), the non-pipelined architecture receives an output every period of time that equals to the sum of the processing time of all stages. In Figure 5-4(b), the pipelined architecture receives an output every period of time after

an initial delay. The length of the period is decided by Stage 2, which has the longest processing time of all stages.

5.3.1 General Block Diagram

The overall hardware architecture of the SIFT processing core is shown in Figure 5-5, which mainly consists of three parts that are arranged into a three-stage pipelined architecture: feature detection, descriptor generation, and descriptor matching. Pipeline stage 1 (feature detection) inputs the 8-bit grayscale pixel stream $I(x, y)$ and outputs the coordinates of detected keypoints $FC(x, y)$ as well as the GMOs of all pixels from the pre-defined scales. In stage 2, a 72-dimention descriptor $Desc(x, y)$ is generated for each keypoint detected in stage 1. In stage 3 (descriptor matching), the keypoint matching is performed based on the descriptors generated from stage 2 and outputs the coordinates of the matched keypoint pairs $MC(x, y)$. The GMOs are buffered using external memory DDR3 on the FPGA board to save on-chip memory.

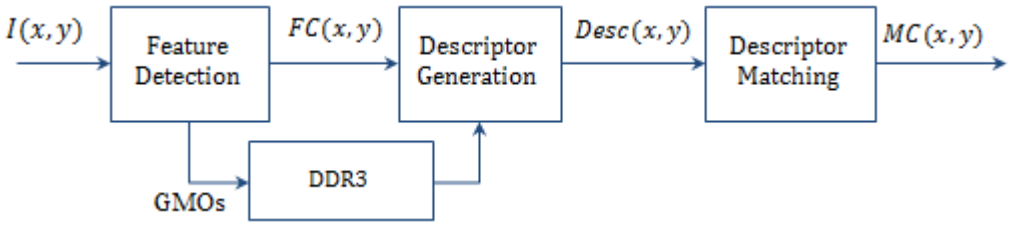


Figure 5-5: Block diagram of the SIFT processing core.

5.3.2 Memory Overview

Carneiro and Jepson [67] have noticed that the number of interest points is around 0.3% of total image size for the state-of-the-art methods developed by Lowe [10] and by Mikolajczyk and Schmid [68]. In this design, the memory is designed under the assumption that the maximum number of stable keypoints is 1,536 for octave 0 and 512 for octave 1, which never overflows in the experiments.

The ping pong buffer management is employed and output of each module is buffered for input to the next pipeline stage so as to pipeline each stage making them processing concurrently. A ping pong buffer is used in a data transfer and contains two identical buffers. While one buffer is receiving data from the previous stage, the other one is being read for the next stage. This type of memory management ensures a real-time processing. In this system, RAM holding input images from the commercial camera is designed as a ping pong buffer to ensure that images can be correctly received while the previous frame is being processed. Similarly, the buffer between individual processing modules is implemented as a ping pong buffer for fully pipelined architecture, such as the buffer holding keypoint information between feature detection module and descriptor generation module. The design of an input image buffer is complicated in that it is simultaneously accessed by the commercial camera, SIFT processing core and USB interface. The architecture will be given in details in Chapter 6 together with the introduction to the camera interface and the USB interface.

5.3.3 Feature Detection

The first part of the SIFT algorithm is the feature detection module which mainly consists of three blocks:

1. The first block in the diagram is the Gaussian scale space and Difference-of-Gaussian (DoG) space construction block that applies Gaussian filter windows of different sizes to the source image to generate a set of smoothed images. Then the subtract operations are applied to adjacent smoothed images to generate the DoG images.
2. The second block is the keypoint detection with stability checking, which is responsible for searching for keypoints from DoG space, refining keypoint locations, and eliminating pixels with low contract or large edge response. Stability checking is important in that pixels with low contrast is sensitive to noise and the difference-of-Gaussian function will have a large response along edges therefore is unstable to small amount of noise.

3. The third block is the GMO calculation, where each pixel is assigned a gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ based on the local image properties in preparation for the descriptor generation.

The overall hardware architecture for feature detection module is shown in Figure 5-6. The pixel stream is input to Block 1 and this block outputs the DoG values and Gaussian smoothed pixels that are buffered as an input to Block 2 and Block 3, respectively. Block 2 is responsible for identifying stable keypoints from DoG space, whereas Block 3 calculates the GMO of pixels from the pre-defined scales.

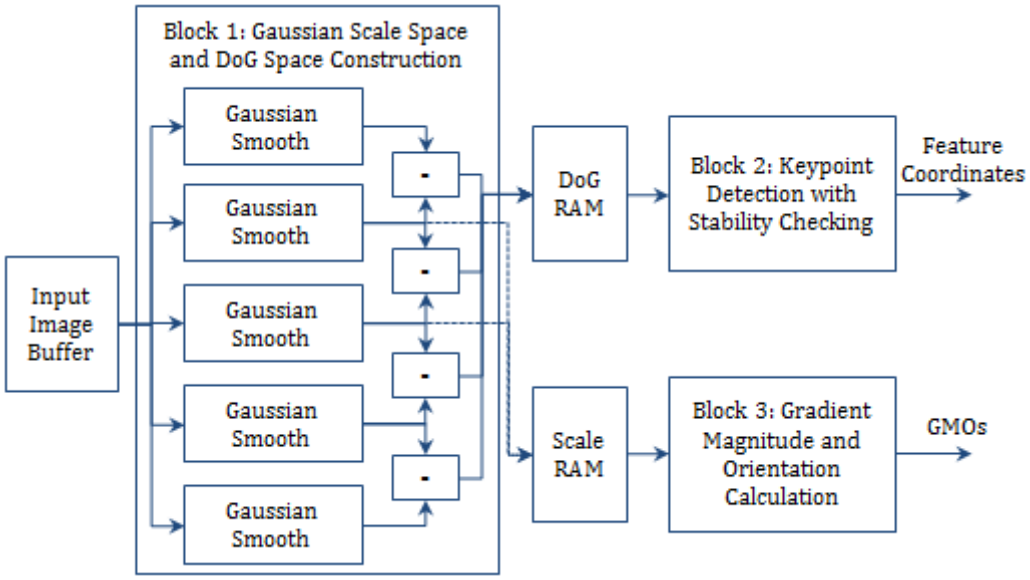


Figure 5-6: Block diagram of feature detection module.

As shown in Figure 5-7, the three blocks are arranged into a three-stage pipelined and partially parallel architecture. Since Block 2 has no data dependency with Block 3, these two blocks are processed in parallel, which cuts the input-output delay by one unit of time when compared with the pipelined architecture. Detailed introduction to Block 1 and Block 3 are presented in the following sections. The hardware architecture of Block 2 is given in Appendix A.

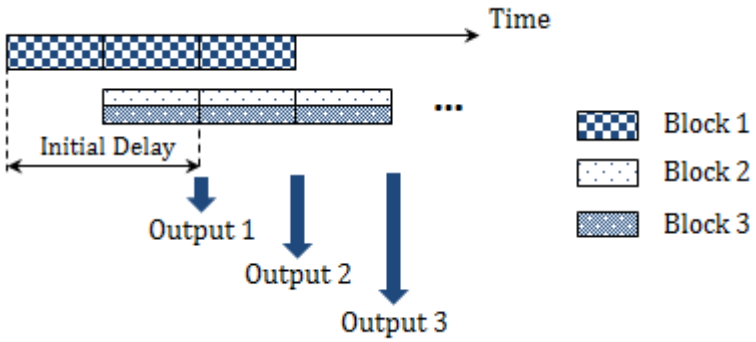


Figure 5-7: The pipelined and partially parallel architecture.

a. SRT Based Multi-Pixel Processing Scheme

In this design, a SRT-based multi-pixel processing method is proposed, with which the time requirement for accessing the pixels for Gaussian scale space construction is reduced.

Processing Time

Figure 5-8(a) shows the source image to be smoothed, where the shadow area represents the pixels located in the boundary region within which the pixels are invalid as a result of the Gaussian smooth process. The rectangular (in red) displayed on the top left corner of the source image represents the region of pixels for smoothing the first valid pixel in the image, which is shown in details in Figure 5-8(b). In Figure 5-8(b), each dot represents a pixel. The dots filled with shadow indicate those invalid pixels that are located in the boundary region, and the red dot in the center of the region represents the first valid pixel to be smoothed. Both the boundary size ($(k_G - 1)/2$) and the size of region ($k_G \cdot k_G$) for each Gaussian filter process are decided by that of the largest Gaussian kernel (k_G) applied. In the exampled given in Figure 5-8(b), k_G is set to 15.

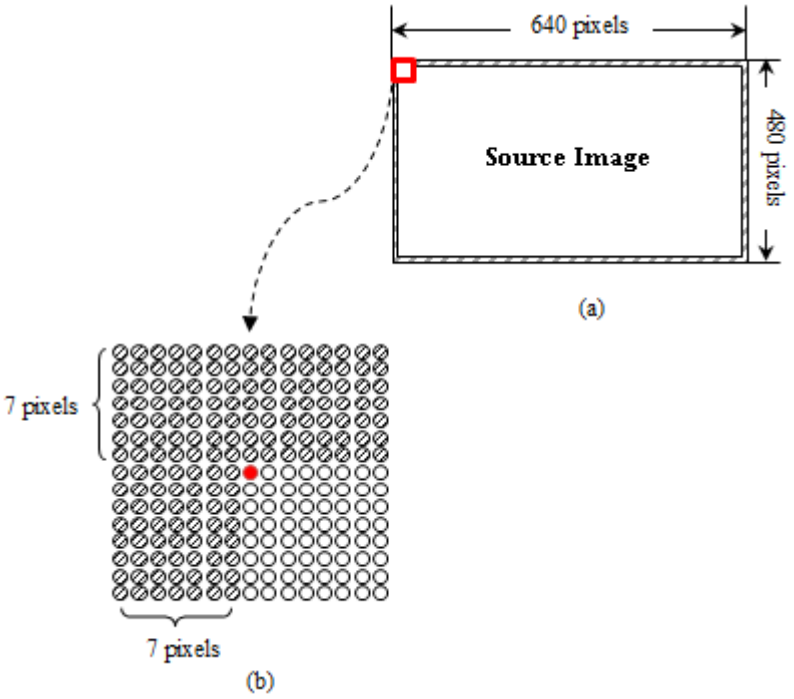


Figure 5-8: Pixels included for Gaussian smooth of the first valid pixel.

Gaussian filter has been quantised to reduce the computation complexity of Gaussian convolution operations. The required processing time for one VGA image is given in (5.1). Different scales are processed in parallel and the RAMs buffering source images are configured as Dual Port RAM (DPRAM) with both read and write accesses to the memory allowed on either port. The BRAM holding input images works with the clock domain of 200MHz, which corresponds to clock cycle of 5ns.

$$TR_{orig} = 5 \cdot \frac{1}{2} \cdot \sum_{i=0}^{1} [(w_i - 2k_G)(h_i - 2k_G)(k_G)^2] \qquad (5.1)$$

where i is the index to octaves. $(w_i - 2k_G)$ and $(h_i - 2k_G)$ represent the number of valid pixels to be smoothed in each row and column of the source image, respectively. ${k_G}^2$ is the number of pixels involved in smoothing one valid pixel independent of the number of scales.

Gaussian scale space can be constructed by smoothing the source image with a large Gaussian kernel for each of the five scales instead of applying multiple successive

Gaussian kernels. Taking advantage of the parallel processing property of FPGA, scale images within the same octave can be generated in parallel by applying Gaussian kernel of different sizes to the source image concurrently. In this case, the memory for buffering intermediate smoothed results using cascade Gaussian filtering is saved, and accessing the source image several times per octave can be avoided. As a result of parallel processing, the number of scales per octave has no effect on TR_{orig}, and TR_{orig} is equal to 405 ms for VGA image. However, the time allowance is no more than 16.7 ms per frame for a frame rate of 60 fps. In this design, the SRT-based multi-pixel processing scheme is proposed for real-time processing. The theoretical time requirement for pixel access is calculated by (5.2).

$$TR_{multi} = 5 \cdot \text{ceil}\left(\frac{k_G + n_{pixel} - 1}{2}\right) \cdot \sum_{i=0}^{1}\left[w_i \cdot ceil\left(\frac{h_i - k_G + 1}{n_{pixel}}\right)\right] \tag{5.2}$$

where n_{pixel} denotes the number of pixels smoothed in parallel. w_i and h_i are the width and height of the input image to each octave. "ceil" indicates round-up to the closest integer. k_G is the size of the largest Gaussian kernel and is set to 31. $\text{ceil}\left(\frac{k_G+n_{pixel}-1}{2}\right)$ is the number of clock cycles (5ns) required to access a column of pixels from the DPRAM.

In this design, the Gaussian scale space consists of two octaves with five scales each. The time consumption of pixel streaming with different number of pixels processed in parallel for VGA image is shown in Table 5-2. The second configuration (n_{pixel}=2) meets the throughput requirement of at least 60 fps and is chosen to demonstrate the efficiency of both the proposed SRT-based image streaming method and the memory solution. Detailed description of the SRT-based multi-pixel processing strategy will be given in next section.

Table 5-2: Time requirement for different number of pixels processed in parallel.

Frame	n_{pixel}	Time requirement (ms/frame)	Achieved throughput (fps)
VGA	1	28.416	35
	2	14.208	70
	3	10.064	99
	4	7.589	131

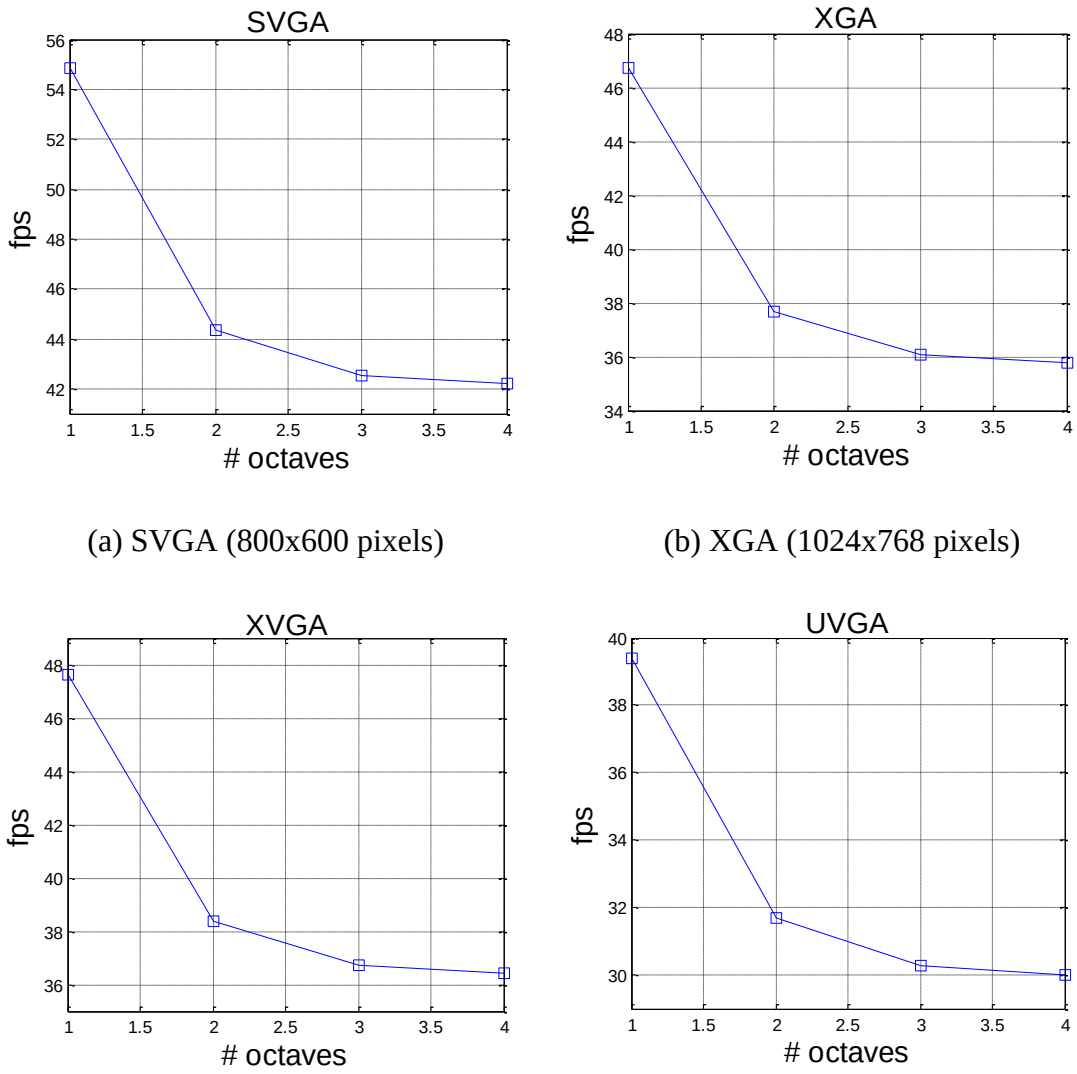
Although the multi-pixel processing method is proposed for VGA sized video, it can be applied to systems with source images of higher resolution for real-time processing. Table 5-3 shows the number of pixels to be processed in parallel for images of higher resolution to achieve real-time performance. The estimated throughput for images of higher resolution is given in Table 5-3.

Table 5-3: Throughput estimation for different frame sizes with multi-pixel processing scheme.

Frame	Resolution (pixels)	n_{pixel}	Time consumption (ms/frame)	Estimated throughput (fps)
SVGA	800x600	2	22.56	44
XGA	1024x768	3	26.55	37
XVGA	1280x800	4	26.06	38
UVGA	1600x1200	7	31.54	31

Although the design is configured to process two octaves with five scales each, larger number of octaves can be processed by making slight modification to the VHDL codes. Because all the octaves are processed in serial and the same processing block is shared, the amount of occupied device will remain almost constant as the number of octaves increases. By using a larger number of octaves, the logic for control the data routing needs to be increased, and also the size of memory blocks

storing the information of detected keypoints. Figure 5-9 shows that the design still achieves real-time despite of the higher image resolution and larger number of octaves.



(a) SVGA (800x600 pixels)

(b) XGA (1024x768 pixels)

(c) XVGA (1280x800 pixels)

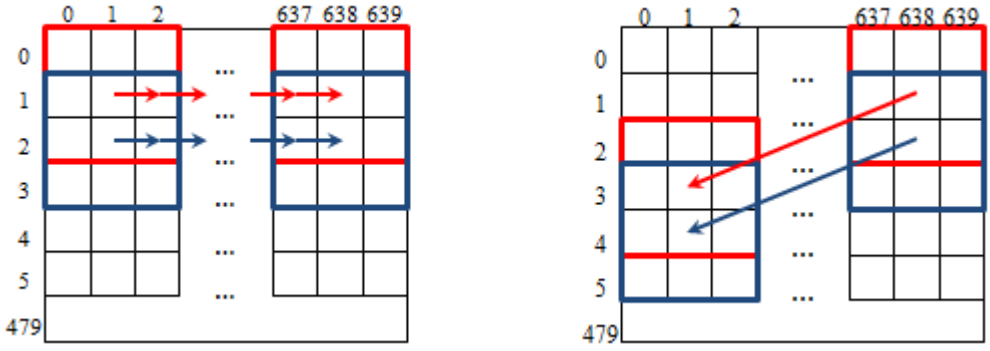
(d) UVGA (1600x1200 pixels)

Figure 5-9: Frame rate and processing time as a function of the number of octaves.

Overview of Gaussian Filter Window Movement

In general, smoothing an image with a Gaussian kernel is equivalent to shifting the filter window over the entire image pixel by pixel. For simplicity without losing

generality, the Gaussian kernel of size 3×3 is used as an example to illustrate the Gaussian filter window movement. With n_{pixel}=2 chosen to parameterise the design, two identical Gaussian filters are employed to smooth the image concurrently. In Figure 5-10, Gaussian filter windows are defined by thick lines and the each square represents a pixel. Figure 5-10(a) and Figure 5-10(b) shows the Gaussian filter window movement in horizontal direction and vertical direction, respectively. The arrows indicate the direction of movement, and the image is scanned in an order from left to right and top to bottom. As shown in Figure 5-10(a), when the first two pixels in the same column have been smoothed, the two identical filter windows move horizontally by one column to smooth the next two pixels. When the filter windows reach the rightmost end of the source image, they return to the leftmost end and move down vertically by two rows to start a new round of horizontal scan, as shown in Figure 5-10(b).



(a) (b)

Figure 5-10: Gaussian window movement over the image in horizontal and vertical direction.

Pixel Streaming Strategy

A SRT-based multi-pixel processing method is proposed, with which the time requirement for pixel access is reduced by sharing pixels between adjacent Gaussian smoothing processes in both horizontal and vertical directions. Each output of a 3×3 Gaussian filter is a function of nine pixels within the window. Without the register,

each pixel must be read nine times as the filter window is scanned through the image. Pixels adjacent horizontally are involved in successive filter processes, so they may be buffered and delayed in registers for sharing. This reduces the number of reads from nine to three pixels for each filter process, with which the accessing time increase linearly with the Gaussian window size instead of exponentially without using registers. A 3×3 filter spans three columns (two previous columns and the current column), and hence the previous two columns can be inherited from the previous filter process and buffered in the register, while a new column of pixels is read in.

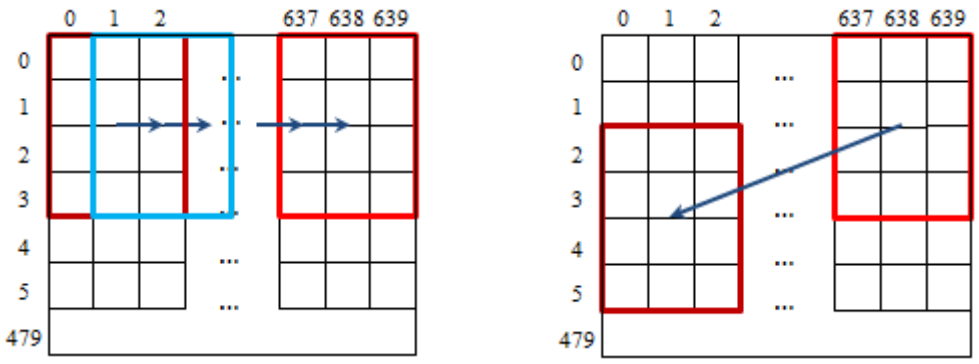


Figure 5-11: Source image streaming at the *process* level.

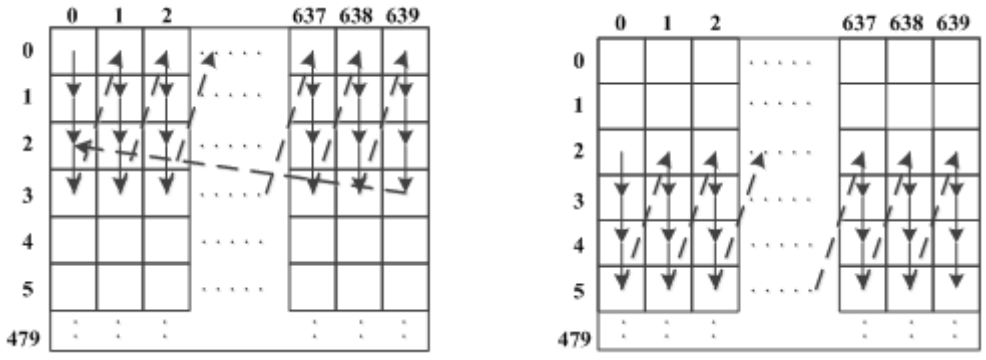
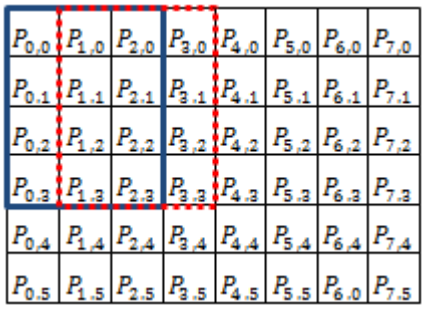
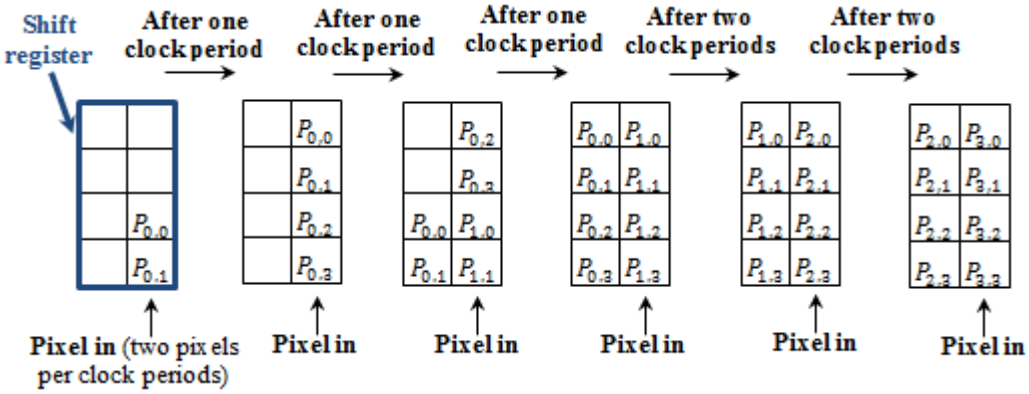


Figure 5-12: Source image streaming at the pixel level.

Figure 5-11 and Figure 5-12 show the source image streaming strategy at different levels. The arrow indicates the streaming path. Smoothing n_{pixel} adjacent pixels in parallel is referred to as a *process*. The pixels involved by each smoothing *process* are defined by the window in thick line, which consists of several contiguous rows of pixels. When it comes to the hardware design, pixels are constantly streamed into a SRT where the Gaussian smooth is performed. The SRT corresponds to the *process* window in Figure 5-11 and is of $(k_G + n_{pixel})$ rows by k_G columns when using conventional 2D Gaussian kernel. The *process* window movement is synchronised with that of Gaussian filter window shown in Figure 5-10. It can be seen from Figure 5-11 that the source image is accessed horizontally at the *process* level with the sequence indicated by the arrow, and vertically at the pixel level as shown in Figure 5-12.



(a) Source Image



(b) Pixel streaming into the SRT

Figure 5-13: The SRT-based image streaming.

Figure 5-13 illustrates the SRT-based pixel streaming method. The SRT holding pixels involved in the Gaussian filter operations is updated dynamically while the pixels are being smoothed. Once a round of Gaussian smooth operation is finished, the SRT is updated with a new column of pixels from the buffer where the source image is located and is ready to start another round of Gaussian smooth operation. Two separate one-dimensional (1D) Gaussian kernels are used instead of the conventional two-dimensional (2D) Gaussian kernel by taking advantage of the linearly separable property of Gaussian kernel, with which the size of the SRT is reduced from $k_G(k_G + n_{pixel})$ to $2(k_G + n_{pixel})$. This strategy is consistent with the multi-pixel streaming method at the pixel level and enables the re-use of intermediate results, which will be discussed in next section. As shown in Figure 5-13(b), pixels are constantly streamed into the left column of the register where the 1D Gaussian smooth is performed in the vertical direction. With two pixels accessed per clock cycle, the left column is updated every ceil$((k_G + 1)/2)$ clock cycles (5ns) for a Gaussian kernel of size k_G.

Gaussian Convolution

In the conventional 2D Gaussian smooth operation, the Gaussian kernel is directly applied to the pixel window and produces a result at the central position of the window in the output image. Figure 5-14 shows the 2D convolution between a pixel window of size 3x3 and a 2D Gaussian kernel of the same size.

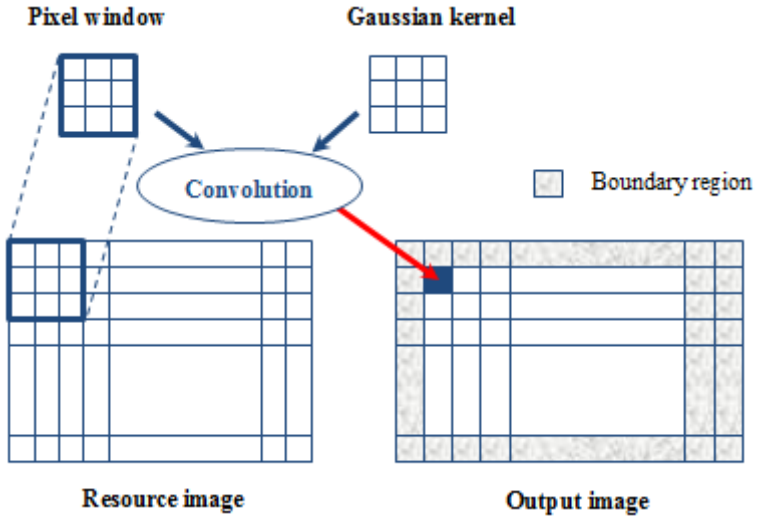
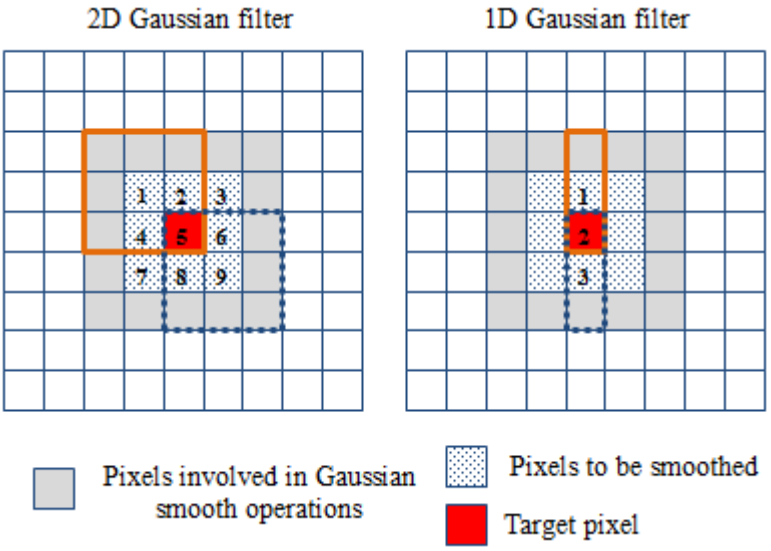


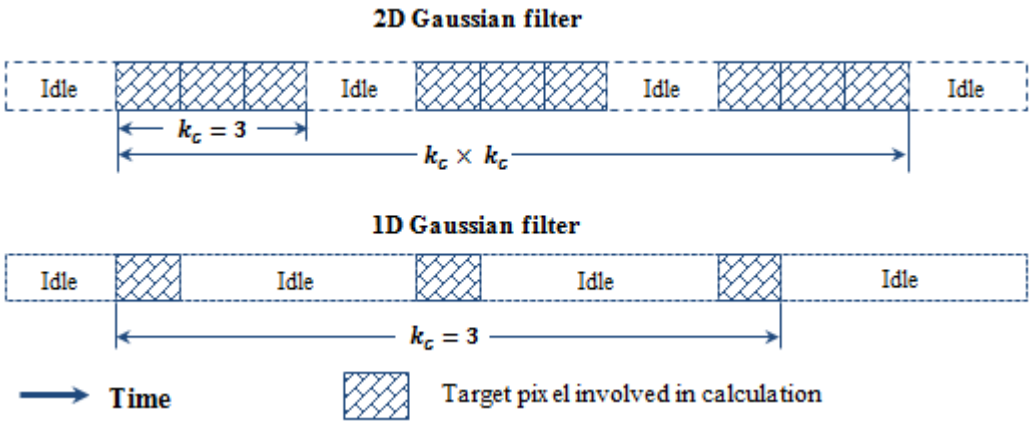
Figure 5-14: Gaussian smooth operation on a window of pixels (3x3) using conventional 2D Gaussian kernel.

The conventional 2D Gaussian smooth operation is ineffective in that each pixel is involved in the Gaussian smooth operation of a region of pixels centered on it, where the region is of size $k_G \times k_G$, as shown in the left image of Figure 5-15(a). To improve the computational efficiency of Gaussian smooth process, conventional 2D Gaussian kernel is substituted by two separate 1D Gaussian kernels by taking advantage of Gaussian kernel's linearly separable property as shown in (5.3).

$$G_{2D}(\mathrm{x},\mathrm{y}) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}\right) = G_{1D}(\mathrm{x}) \cdot G_{1D}(\mathrm{y}) \qquad (5.3)$$



(a)



(b)

Figure 5-15: Comparison of 2D and 1D Gaussian convolution, in terms of computation efficiency at the pixel level.

The 1D Gaussian smooth consists of two stages. In the first stage, a 1D kernel is used to smooth the image in the vertical/horizontal direction. In the second stage, another 1D kernel is used to smooth in the perpendicular direction. In Figure 5-15(a), the filter window in thick lines and dashed lines represents the first and the last Gaussian smooth operation in which the target pixel is involved within its neighborhood, respectively. As shown in Figure 5-15(b), the target pixel is processed $k_G{}^2$ times when using 2D Gaussian kernel. However, each pixel only needs to be processed k_G times by using 1D Gaussian kernel instead, which benefits from the intermediate results usage of the 1D Gaussian smooth in the first direction.

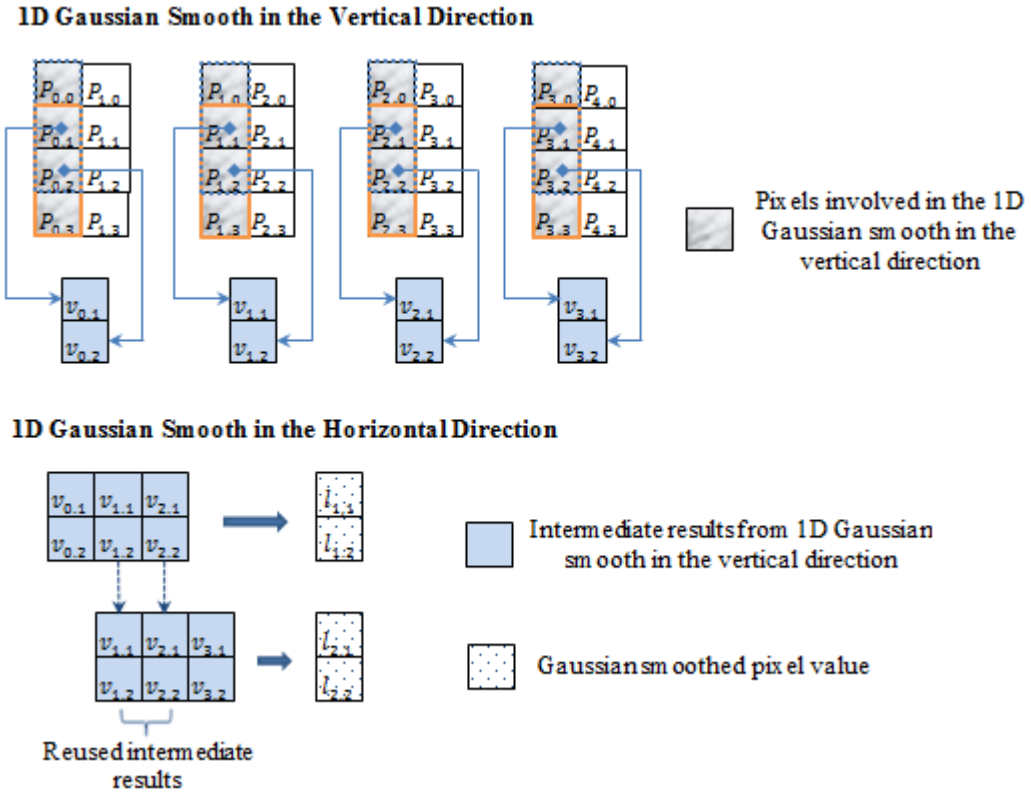


Figure 5-16: Block diagram of the SRT-based 1D Gaussian smooth with intermediate results re-used.

Figure 5-16 shows the diagram of the 1D Gaussian smooth with the re-use of intermediate results. Since the source image is streamed into the SRT vertically, the 1D Gaussian smooth is first performed in the vertical direction, then in the horizontal

direction. Intermediate results of 1D vertical Gaussian smooth are continuously shifted into and out of a register, where the 1D Gaussian smooth is performed in the horizontal direction. For the 1D Gaussian kernel of size k_G, the smooth result from the vertical direction can be re-used for the 1D Gaussian smooth in the horizontal direction for the following $(k_G - 1)$ *processes*, which reduces the computation complexity of the system. As a result, the data in the SRT holding intermediate results from 1D Gaussian smooth in the vertical direction falls into two categories below:

1) The data in the leftmost $(k_G - 1)$ columns are inherited from previous 1D Gaussian smooth in the vertical direction;
2) The data in the rightmost column is new and is generated by applying 1D Gaussian smooth in the vertical direction to the rightmost column of pixels in the current *process* window.

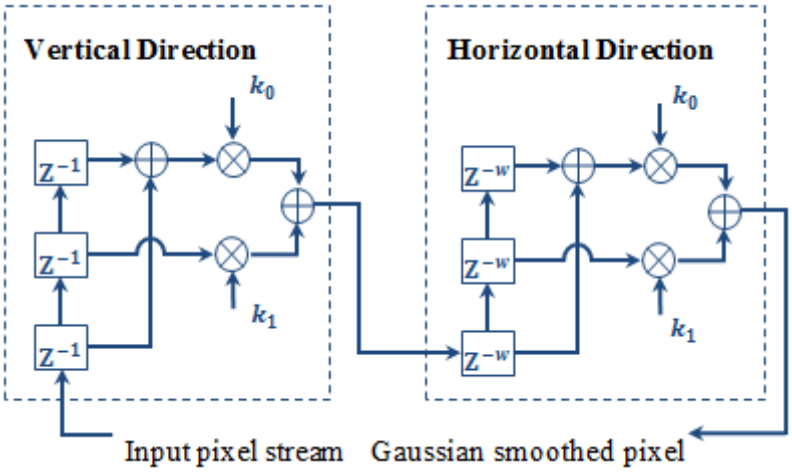


Figure 5-17: Pipelined architecture for the Gaussian smooth process with 1D Gaussian kernel of size $k_G = 3$.

To achieve the maximum throughput, the pixel streaming process and the Gaussian convolution process are arranged into a two-stage pipelined architecture. In Figure 5-17, the left side shows the 1D Gaussian convolution in the vertical direction and the right side shows the convolution operation in the horizontal direction. In this case,

the source image is continuously streamed into the computation module. Figure 5-18 shows the typical timing diagram of the Gaussian convolution using 1D Gaussian kernel. Because the SRT holding pixels for 1D Gaussian smooth is updated every $\frac{k_G+1}{2}$ clock cycles of 200 MHz, two Gaussian smoothed pixels that are processed in parallel can be obtained every $\frac{k_G+1}{4}$ clock cycles of 100 MHz after an initial delay.

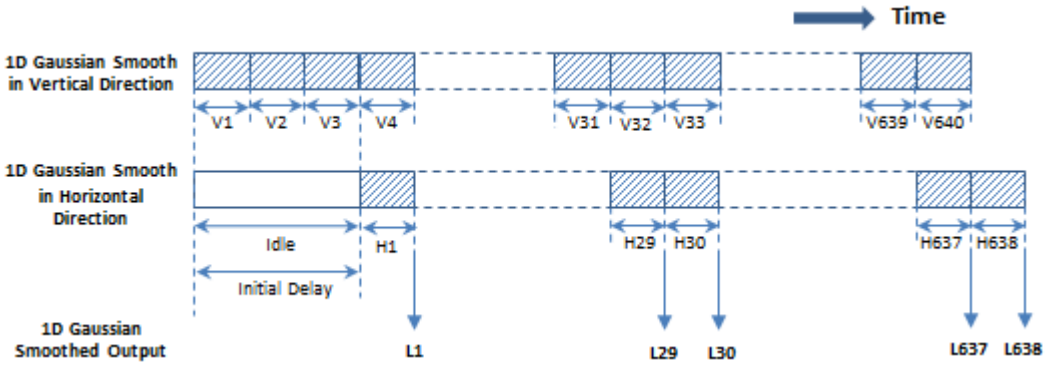


Figure 5-18: Timing diagram for 1G Gaussian convolution with 1D Gaussian kernel of size k_G=3.

Each 2D Gaussian smooth requires ${k_G}^2$ multiplication-accumulation (MAC) operations. The number of MAC operations for a 1D Gaussian convolution to obtain a result is k_G. Therefore, it requires $2k_G$ MAC operations to generate a smoothed pixel that is equivalent to a 2D convolution output. The computational advantage of the non-separable 2D convolution against the separable 1D convolution is ${k_G}^2/2k_G$. For Gaussian kernel of size k_G=31, the use of 1D Gaussian convolution introduces a reduction in the number of MAC operations by a factor of 15.5, which indicates a reduction of up to 15.5 times in the use of device area for these operations. The total number of MAC operations to be performed on an M × N sized image to construct the Gaussian scale space of O octaves and S scales using 2D and 1D are given in (5.4) and (5.5), respectively.

$$\mathrm{MAC}_{Gauss2D} = \sum_{i=0}^{O-1} \frac{MN}{4^i} {k_G}^2 S \tag{5.4}$$

$$\mathrm{MAC}_{Gauss1D} = \sum_{i=0}^{O-1} \frac{MN}{4^i} 2k_G S \tag{5.5}$$

The computational efficiency is further improved by taking advantage of the symmetric property of 1D Gaussian kernel. As shown in Figure 5-17, pixels sharing the same weighting factor are added up before applying multiplication operations. It can be seen from (5.4) and (5.6) that the computational cost increases linearly with kernel size instead of exponentially as the 2D convolution does, which has greatly reduced the number of operations for Gaussian smooth and further the device area.

$$\mathrm{MAC}_{Gauss1D}{}' = \sum_{i=0}^{O-1} \frac{MN}{4^i} (k_G + 1) S \tag{5.6}$$

Memory Solution

Initially, the memory requirement for buffering Gaussian smoothed pixels (MR_{scale}) of one scale and DoG values (MR_{DoG}) are given in (5.7) and (5.8), respectively.

$$MR_{scale} = e[(w_0 - 2b)(h_0 - 2b)] \tag{5.7}$$

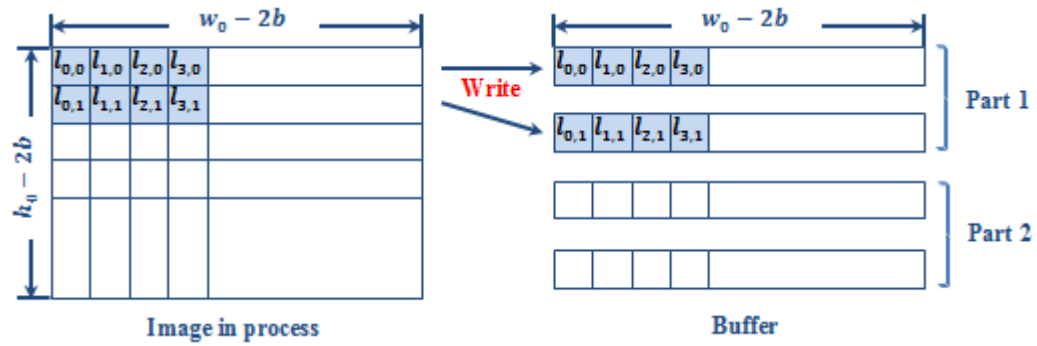
$$MR_{DoG} = 4l[(w_0 - 2b)(h_0 - 2b)] \tag{5.8}$$

where e and l are the word length of a Gaussian smoothed pixel and a DoG value, respectively. b is equal to $(k_G - 1)/2$ and is the size of the boundary region within which both the filtered pixels and the DoG values are unavailable due to the nature of Gaussian smooth.

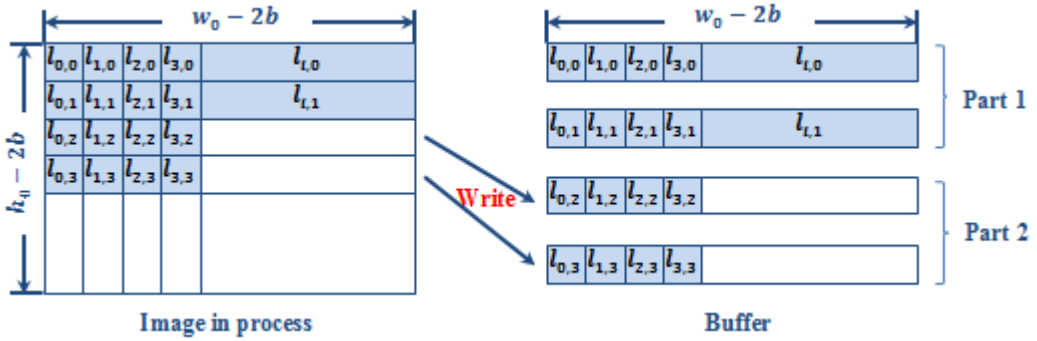
There is no need to buffer all the smoothed pixels of the entire scale, since the GMO calculation of a pixel is only related to its four neighbouring pixels. This is the same case with that of DoGs, since the keypoint detection is only related to the 26 neighbouring pixels in a 3×3×3 region. Figure 5-19 shows an efficient memory

solution to buffer Gaussian smoothed image, named as *rotating buffer*. DPRAM acts as the buffer for efficient memory access. The data depth and width of the RAM are $4(w_0 - 2b)$ and e, respectively, where $(w_0 - 2b)$ is the number of valid pixels per row for octave 0. The RAM for DoG values is of the same depth with that for smoothed pixels, but the data width is $4l$ with four DoG values of a pixel concatenated and saved as one data for fast access. Since the same access strategy is used for both Gaussian smoothed pixels and DoG values, the *rotating buffer* for Gaussian smoothed pixels is used as an example to describe the efficiency of the proposed method.

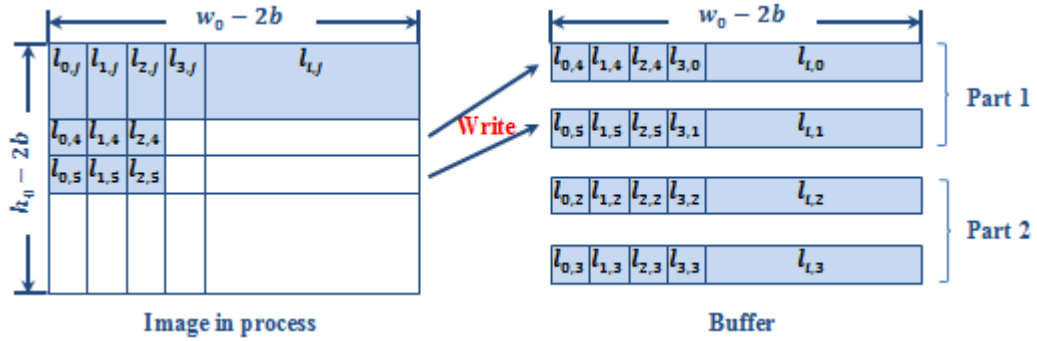
In Figure 5-19, each square in the image being processed represents a pixel (the boundary region is not shown). The squares in colour are the pixels that have been smoothed. i is the index to the columns of the image and is in range [0, w_0-2b-1], and j is the index to the rows of the image and is in range [0, h_0-2b-1]. In the right image of Figure 5-19, each square in the buffer holds a Gaussian smoothed pixel. The buffer is divided into two parts with each part having two rows. The buffer is accessed in a way that one part is being written while the other part is being read. Figure 5-19(a) shows that the Gaussian smoothed pixels are written to Part 1 of the buffer when the source image is being smoothed in the first round of scan. When Part 1 has been filled with smoothed pixel values from the first round of scan, the following two rows of smoothed pixels from the second round of scan are mapped to Part 2, as shown in Figure 5-19(b). Figure 5-19(c) shows that when it comes to the third round of scan, the smoothed pixels are written back to Part 1, overwriting the pixel values from the first round of horizontal scan, and so forth.



(a)

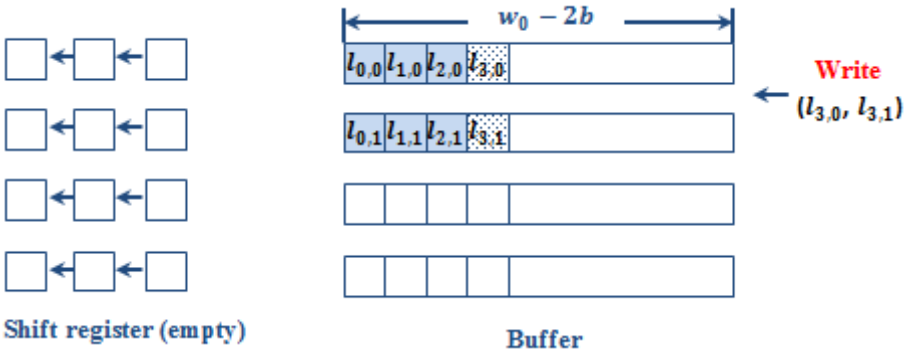


(b)

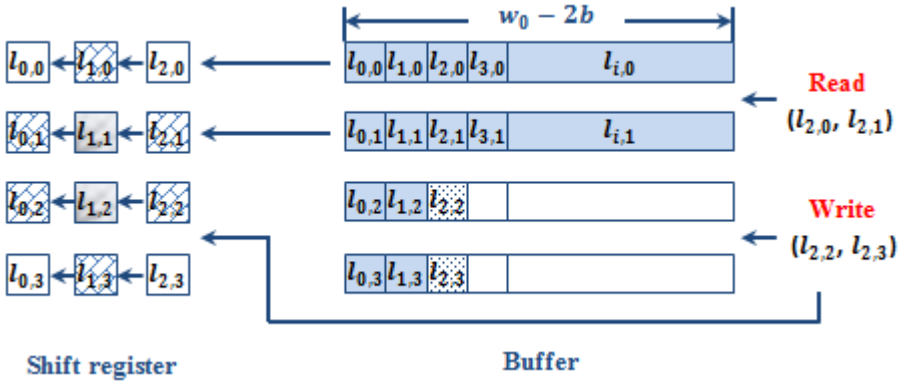


(c)

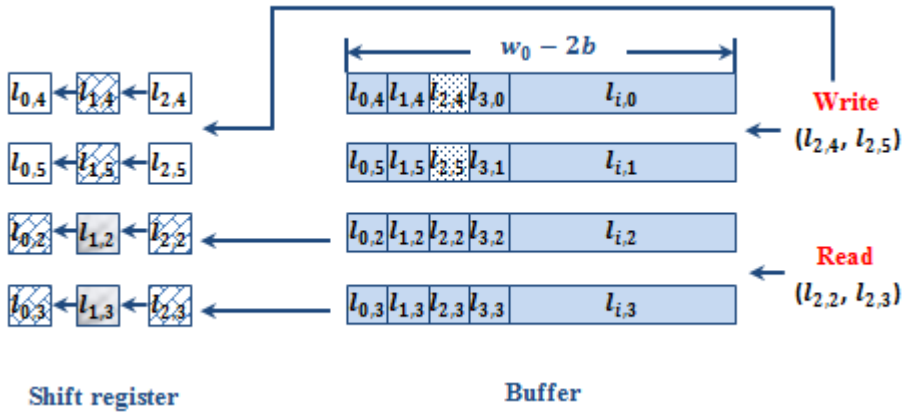
Figure 5-19: The *Rotating buffer* based memory solution for Gaussian smoothed pixels with $n_{pixel} = 2$.



(a)



(b)



(c)

Pixels that have been smoothed

Pixels being smoothed

Pixels who's GMOs are being calculated

Pixels involved in the GMO calculation

Figure 5-20: The SRT-based data access for GMO calculation with $n_{pixel} = 2$.

As shown in Figure 5-20, a SRT of four rows by three columns is used to hold the Gaussian smoothed pixels for GMO calculation while the source image is being processed. The size of the SRT is decided by the number of pixels smoothed in parallel, and is of size $(n_{pixel} + 2) \times 3$. Figure 5-20(a) shows that the SRT remains empty before Part 1 of the buffer has been filled with Gaussian smoothed pixels from the first round of scan. As shown in Figure 5-20(b) and Figure 5-20(c), when it comes to the following rounds of scan, smoothed pixels are continuously streamed into and out of the SRT for GMO calculation and the smoothed pixels are accessed in two ways:

1) The two rows of smoothed pixels with higher physical level in the smoothed image are accessed from the *rotating buffer*.
2) The other two rows of smoothed pixels with lower physical level in the smoothed image are continuously shifted into the SRT while the same rows of source image are being smoothed.

Therefore, of the total four newly updated scaled pixels of each column in the SRT, two are retrieved from the *rotating buffer* and the other two are from the pixels being smoothed. By taking advantage of the DPRAM provided by FPGAs, the two parts of buffer can be accessed simultaneously and independently.

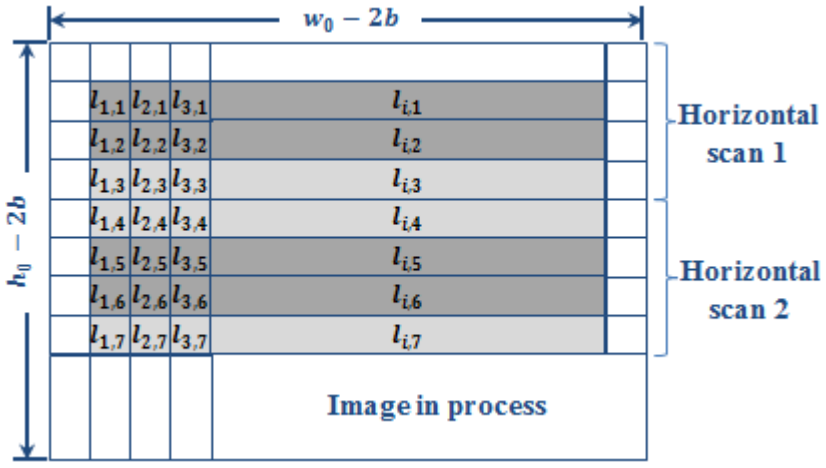
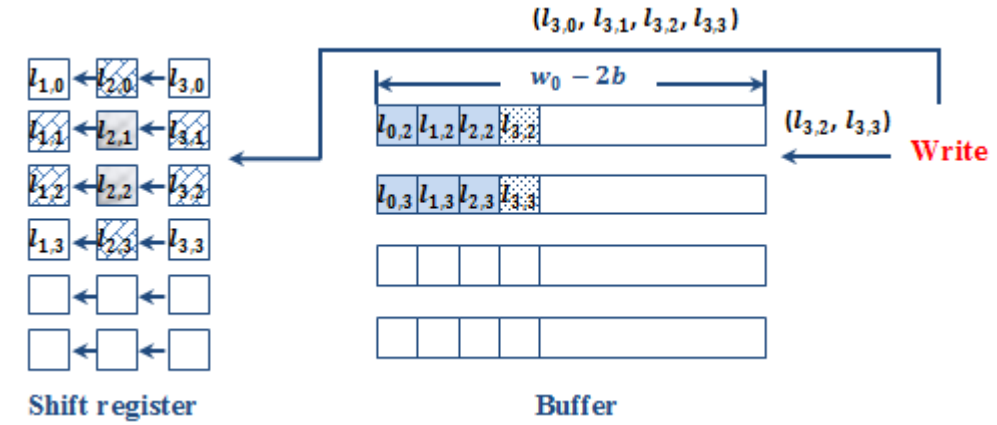
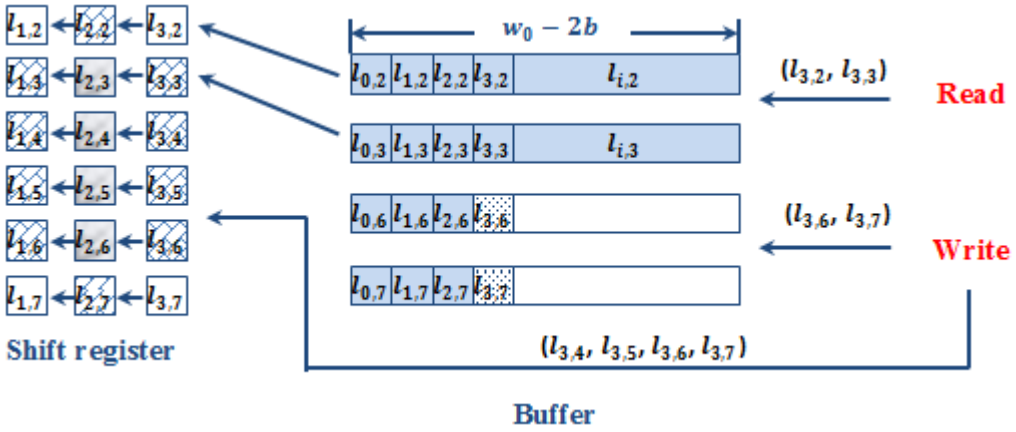


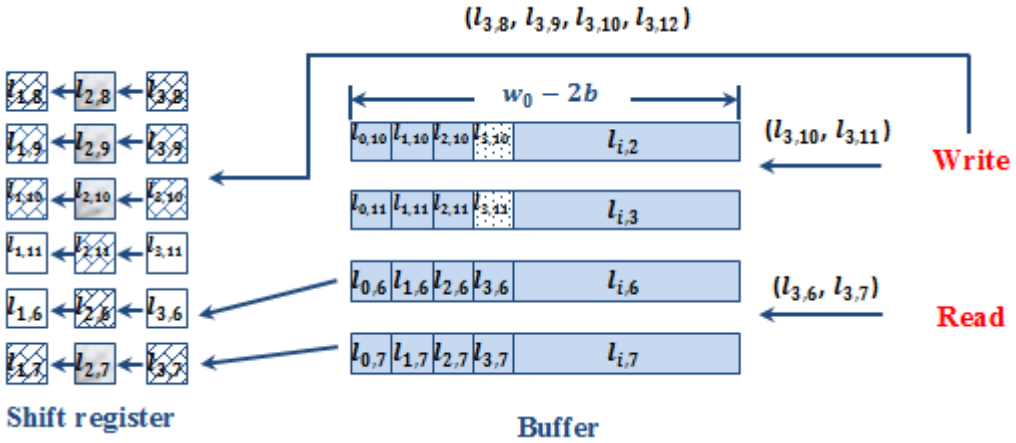
Figure 5-21: Gaussian smoothed image with $n_{pixel} = 4$.



(a)



(b)



(c)

Pixels that have been smoothed

Pixels being smoothed

Pixels who's GMOs are being calculated

Pixels involved in the GMO calculation

Figure 5-22: The SRT-based data access for GMO calculation with $n_{pixel} = 4$.

It should be noted that the *rotating buffer* for either smoothed pixels or DoG values is of size $4 \times (w_0 - 2b)$, which remains constant and is independent of the number of pixels processed in parallel. When n_{pixel}=4, for example, four rows of pixels (row 0 to row 3, row 5 to row 8, and so forth) are smoothed in parallel. As shown in Figure 5-21, GMO calculation of pixels in dark grey (row 1, row 2) is only related to pixels smoothed in the same round of scan (row 0 to row 3). However, GMO calculation of pixels in light grey (row 3, row 4) involves the smoothed pixels of the adjacent round of scan (row 2, row 5). Therefore, the buffer only needs to hold the smoothed pixels of the last two rows of each round of scan, independent of the number of pixels processed in parallel. As shown in Figure 5-22, when multiple pixels ($n_{pixel} > 2$) are smoothed in parallel, the size of *rotating buffer* is constant with the increase of n_{pixel} and stays the same with that of $n_{pixel} = 2$, but the size of registers increases from four rows to $(n_{pixel} + 2)$ rows.

The memory requirement for Gaussian smoothed pixels (MR'_{scale}) of a scale and DoG values (MR'_{DoG}) are reduced from (5.7) and (5.8) to (5.9) and (5.10), respectively.

$$MR'_{scale} = e \cdot [4(w_0 - 2b)] \tag{5.9}$$

$$MR'_{DoG} = 4l \cdot [4(w_0 - 2b)] \tag{5.10}$$

where e and l are the word length of a Gaussian smoothed pixel and a DoG value, respectively. It can be seen from (5.9) and (5.10) that the memory consumption is independent of n_{pixel}.

Although the design is proposed for VGA images, it can be applied to images of higher resolution. Table 5-4 summarises the memory requirement for images of different sizes, where the buffer size for DoG values is estimated with five scales per octave. With both buffers shared between octaves, the memory requirement is independent of the selected number of octaves.

Table 5-4: Memory requirement of Gaussian smoothed pixels and DoGs for images with different sizes.

Source image	Image resolution (pixels)	Memory requirement	
		Gaussian smooth (Kbits/scale)	DoG (Kbits)
QVGA	320x240	$1.20e$ (1)	$4.78l$ (2)
VGA	640x480	$2.45e$	$9.78l$
SVGA	800x600	$3.07e$	$12.28l$
XGA	1024x768	$3.95e$	$15.78l$
WXGA	1280x800	$4.95e$	$19.78l$
UXGA	1600x1200	$6.20e$	$24.78l$

(1) e is the word length of a Gaussian smoothed pixel.

(2) l is the word length of a DoG value.

b. Gradient Magnitude and Orientation Calculation

Considering that the GMO computation of a pixel is only related to its four adjacent smoothed pixels in the same scale, they can be computed in parallel with Gaussian smooth with only a negligible initial delay. The block diagram of the proposed approximation based architecture for GMO computation is shown in Figure 5-23.

G_{x_sign} and G_{y_sign} are defined in (5.11) and (5.12), respectively. θ_{temp} represents the gradient orientation in the first quadrant. With the orientation quantised to integers in range 0 to 35, θ_{temp} is integer in range 0 to 8.

$$G_{x_{\text{sign}}} = \begin{cases} 0, & \mathrm{L(x+1,y)} \geq \mathrm{L(x-1,y)} \\ 1, & \mathrm{L(x+1,y)} < L(\mathrm{x-1,y}) \end{cases} \tag{5.11}$$

$$G_{y\_\text{sign}} = \begin{cases} 0, & \mathrm{L(x,y+1)} \geq \mathrm{L(x,y-1)} \\ 1, & \mathrm{L(x,y+1)} < L(x,y-1) \end{cases} \tag{5.12}$$

The approximation based strategy takes only four clock cycles to calculate a pixel orientation.

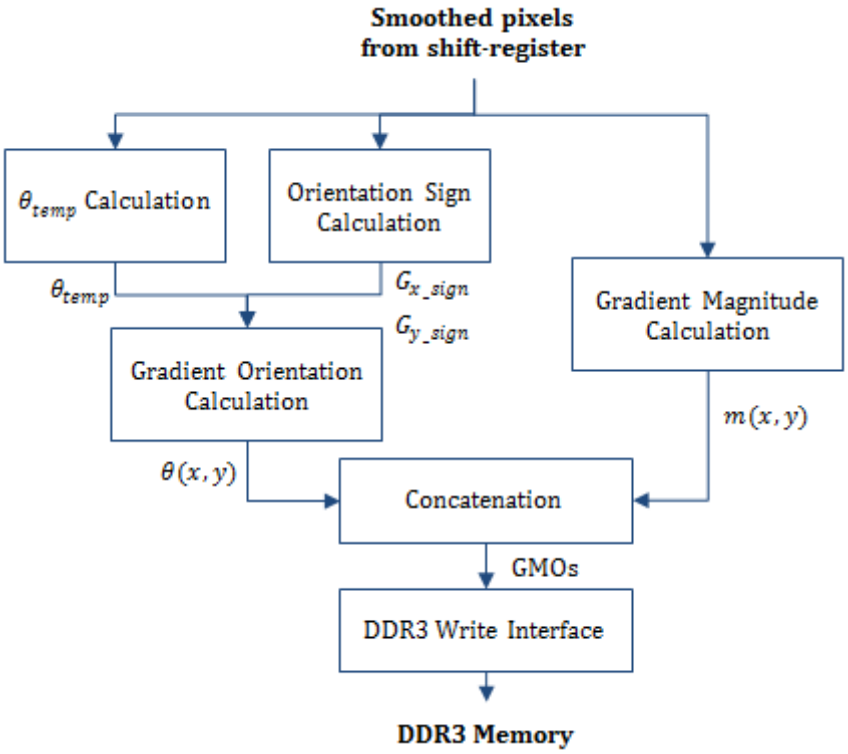


Figure 5-23: Block diagram of the approximation based GMO calculation.

Memory Solution

The memory requirement (MR_{GMO}) for GMOs is defined in (5.13).

$$MR_{GMO} = b_{GMO} \cdot N_{scale} \cdot \sum_{i=0}^{1} [(w_i - 2b - 2)(h_i - 2b - 2)] \qquad (5.13)$$

where b_{GMO} is the bits number per GMO. N_{scale} is the number of pre-selected scales per octave for GMO calculation.

It has been studied in Chapter 4 that gradient magnitude and orientation are represented by 10 bits and 6 bits, respectively. With two scales per octave selected, the required memory size is MR_{GMO}=1.27 Mbytes, which is too large for many hardware devices to afford. To tackle this problem, a DDR3 based memory solution is proposed in this thesis to provide up to 512 Mbytes off-chip memory. The Xilinx EDK development tools provide parameterisable Xilinx Multi-Port Memory

Controller (MPMC) [69], which offers access to DDR3 from one to eight independent ports. Each port can be chosen from a set of Personality Interface Modules (PIMs). In this design, MPMC is configured with two Native Port Interface (NPI) PIMs, which support configurable data width of 32 bits or 64 bits on each port. One is used to write calculated GMOs to DDR3, and the other is used to read GMOs from DDR3 for descriptor generation. As shown in Figure 5-24, four sets of GMOs are concatenated and sent as a single data to make full use of the data width of 64 bits. The throughput requirement (TPR_{GMOWr}) for NPI write port is defined below.

$$TPR_{GMOWr} = f \cdot MR_{GMO} \tag{5.14}$$

where f is the system frame rate.

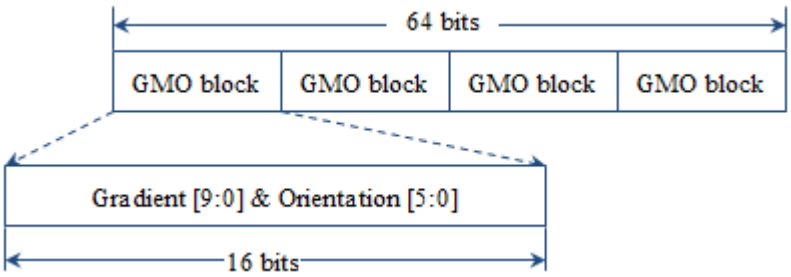


Figure 5-24: Data concatenation for GMO.

The throughput requirement (TPR_{GMOWr}) is 76.1 Mbytes/s with an overall frame rate of 60 fps. It can be seen from Table 5-5 that NPI write port with 32-word burst data transfer meets the throughput requirement.

In Table 5-5, latency on writes is not characterised because MPMC allows write data to be pushed in before or after the address request. It should be noticed that the throughput listed in Table 5-5 is the maximum total data throughput. The throughput increases with the burst size of the transfer data, so the 64-word burst offers the highest maximum bandwidth but might increase the delay on other ports. Therefore, the 32-word burst data transfer is used for DDR3 memory access with priority given to NPI read port to constantly feed GMOs into the descriptor generation module

without interruption, which will be discussed later in section 5.3.4. The detailed configuration of the NPI PIM Write Interface is given in Appendix B.

Table 5-5: MPMC port latency and theoretical throughput for Virtex-6 FPGA

	Port A	**Port B**
Port Type	NPI	NPI
Operation	Write	Read
Data Width	64 bits	64 bits
Data Transfer Type	32-word burst	32-word burst
Initial Transaction Latency (MPMC_Clk0)	N/A	30
Maximum Total Data Throughput (Mbytes/s)	1,143	1,408

5.3.4 Descriptor Generation

The second part of the SIFT algorithm is the Descriptor Generation Module (DGM), where each keypoint is described using a gradient-orientation histogram. The overall hardware architecture of the descriptor generation module is shown in Figure 5-25, which mainly consists of six sub-modules:

1. Gaussian Weighting Factor Controller, which is Look-up Table (LUT) based and provides Gaussian weighting factors for both Principal Orientation Calculation and 36-bin Histogram Generation.
2. Principal Orientation Calculation. This sub-module inputs the GMOs from DDR3 and outputs the principal orientation of the keypoint by weighting and accumulating pixels within the local region.
3. Centre Coordinate Calculation. In this sub-module, the centre coordinates of eight surrounding sub-regions are calculated based on the principal orientation, with which the locations of the surrounding sub-regions are fixed for the orientation invariance of the sub-region arrangement.

4. 36-bin Histogram Generation. By assigning a consistent orientation to each keypoint, each sub-region within the local region is described by using a 36-bin histogram that is represented relative to the principal orientation and therefore achieving rotation invariance. The orientation histogram has 36 bins covering 360 degrees of orientation.
5. Linear Interpolation. The 36-bin histogram is interpolated into 8-bin histogram by distributing the value of each histogram into its adjacent histogram bins so as to avoid abrupt changes in the descriptor as a result of a sample shifts from being within one histogram to another.
6. Descriptor Normalisation. The descriptor is normalised twice to reduce the effects of illumination changes, such as image contrast, and to reduce the influence of large gradient magnitudes.

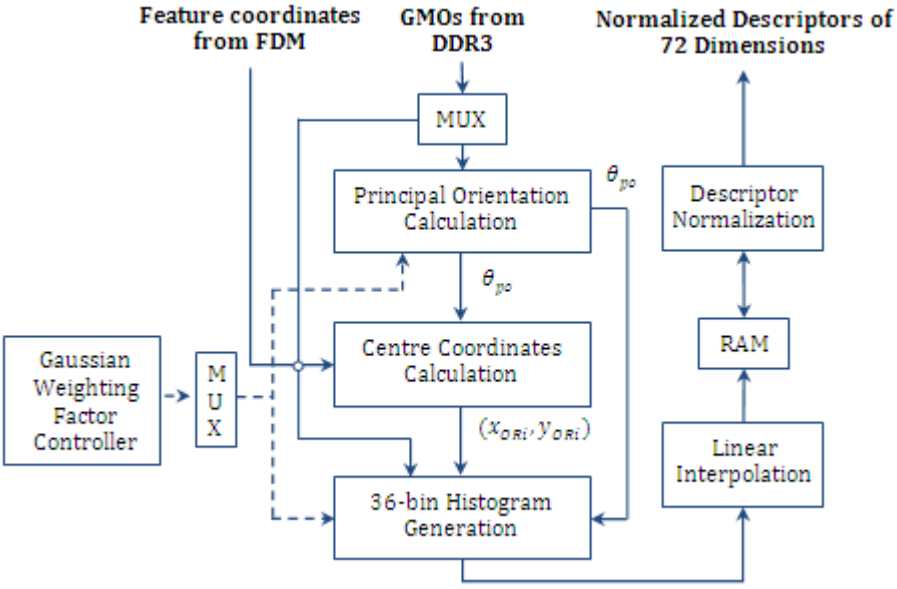


Figure 5-25: Block diagram of descriptor generation module, where FDM stands for the Feature Detection Module.

Taking advantage of the parallel processing property of hardware devices, the six sub-modules are arranged into a five-stage pipelined and parallel processing architecture. It can be seen from Figure 5-25 that Stage 1 inputs the GMOs and the

weighting factors retrieved from the Gaussian weighting factor controller, and outputs the principal orientation, which is then passed to Stage 2 and Stage 3. In Stage 2, center coordinates of eight surrounding sub-regions are computed and fed into Stage 3. Then the 36-bin histogram of each sub-region is generated in Stage 3 with the weighting factors retrieved from Gaussian weighting factor controller. In Stage 4, 36-bin histograms are interpolated into 8-bin histograms, which are normalised in the last stage to generate the final descriptor of 72 dimensions. Detailed introduction to the partition based memory access scheme, the SRT (Shift RegisTer) based reconfigurable divider and the SRT based square root calculator are presented in this section. Hardware architecture of the descriptor generation module is given in Appendix C.

a. Memory Access Scheme

DDR3 is used as the buffer for GMOs (Gradient Magnitude and Orientations). Although NPI PIM read port supports theoretical throughput of up to 1,408 Mbytes/s, it is not large enough if the DDR3 is used directly as the input to DGM without an efficient memory access strategy.

Memory Throughput Analysis

Figure 5-26 shows the sub-regions arrangement. Each sub-region is defined as a rectangle, because it is difficult to define circular sub-regions when accessing pixels from the memory and processing sub-regions to generate gradient histogram.

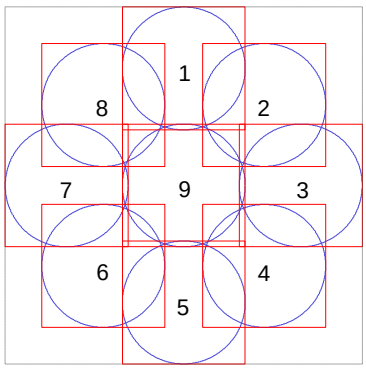


Figure 5-26: Rectangular sub-regions arrangement with overlap.

The square in dashed line indicates the local region centred on the keypoint. The data (GMOs) in the overlapped region have to be repeatedly accessed if each sub-region is accessed independently from the memory, which largely increases the throughput requirement for NPI PIM read interface of MPMC. Therefore, it is more efficient if the entire local region is accessed as a whole from DDR3 and buffered as input to the descriptor generation module. In this case, the throughput requirement (TPR_{NPIRd}) of the NPI read port is defined by (5.15).

$$TPR_{NPIRd} = n_{64bits} \cdot d_l \cdot n_{desc} \cdot f \cdot 64bits \tag{5.15}$$

with

$$n_{64bits} = \text{ceil}\left(\frac{d_l}{4}\right)$$

where d_l is the diameter of the local region. n_{64bits} is the number of 64-bit data to be accessed from the memory for each row of the local region. n_{desc} is the number of descriptors to be generated per frame. f is the system frame rate. The throughput requirement is in range 412.9 Mbytes/s to 1011.7 Mbytes/s with 2,000 keypoints per frames, which is proportional to the size of the sub-region, the number of keypoints to be described, and the frame rate.

The throughput of the NPI PIM read port increases with data burst size, but might increase the delay on other ports. With both ports configured as 32-word burst transfer, neither of them is able to achieve the theoretical throughput given in Table 5-5. It has been obtained from experiments that with 32-word burst data transfer on both NPI PIM port, it requires on the average 46 MPMC clock cycles (5ns) for each read transaction, which corresponds to a throughput of approximately 530.7 Mbytes/s and is not high enough to meet the throughput requirement of the design. To tackle this problem, an efficient memory access strategy is required.

In this thesis, a partition-based memory access scheme is proposed to reduce the throughput requirement increased by repeatedly accessing pixel values within the overlapped area shared between adjacent local regions. With the new memory access strategy employed, the throughput requirement (TPR'_{DDR3}) is defined in (5.16),

which is no longer related to the number of keypoints and is less dependent on the size of sub-regions.

$$TPR'_{DDR3} = f \cdot \sum_{i=0}^{1} \sum_{j=2}^{3} TPR_{si} \tag{5.16}$$

with

$$TPR_{si} = n_{partition} \cdot n_{row_{si}} \cdot n_{64bits} \cdot 64bits$$

where

$$n_{row_{si}} = \text{ceil}\left(\frac{(h_i - 2b - 2) + (n_{partition} - 1) \cdot n_{overlap_{si}}}{n_{partition}}\right)$$

$$n_{64bits} = \text{ceil}\left(\frac{w_i - 2b - 2}{4}\right)$$

j is the index to the two pre-defined scales (scale2 and scale3) and *i* is the index to octaves. $n_{partition}$ is the number of partitions of each octave and $n_{row_{si}}$ is the number of rows per partition. $n_{overlap_{si}}$ is the overlapped rows shared by the local region of keypoints in adjacent rows and is equal to 40 and 64 for scale2 and scale3, respectively.

The throughput and memory requirement of the partition-based solution with different partition sizes are given in

Table 5-6. The time consumption per partition is calculated based on the experimental result, which is 46 MPMC clock cycles per 32-word burst read transaction. The buffer size is closely related to the number of partitions, which decides the number of rows of GMOs to be buffered for each partition. Six partitions for octave 0 and one partition for octave 1 are chosen with the compromise made between throughput and memory requirement, with which the throughput requirement is reduced significantly.

Table 5-6: DDR3 throughput requirement of partition-based memory access solution with different partition size (scale2/scale3).

Octave	Number of partitions	Rows of GMOs (per partition)	Memory requirement (Mbytes)	DDR3 throughput requirement (Mbytes/s)
0	2	244/256	0.66/0.74	33.96/35.63
	3	176/192	0.50/0.59	36.74/40.08
	4	142/160	0.42/0.51	39.52/44.53
	5	122/140	0.38/0.47	42.44/48.71
	6	108/128	0.34/0.44	45.09/53.44
	7	100/118	0.32/0.42	48.71/57.47
	8	92/112	0.31/0.40	51.22/62.34
1	1	208/208	0.23/0.23	6.86/6.86

Partition-based Memory Access Solution

This section describes in details the proposed partition-based memory access solution that is developed to reduce the throughput requirement of the NPI PIM read port. Figure 5-27 shows the block diagram of the memory access solution to NPI PIM read interface. NPI PIM Read Unit fetches GMOs from DDR3 and sends data to DGM through Multiplexing Controller. Read Interface Controller is designed to deal with enable signal ($ReadStart$) and status indicators ($GMOReady_{Oct0}[5:0]$, $GMOReady_{Oct1}$, $DGStart_{Oct0}[5:0]$,$DGStart_{Oct1}$, $DGFinish_{Oct0}[5:0]$, $DGFinish_{Oct1}$) to control the processing procedures of the NPI PIM Read Unit and the Descriptor Generation Module to make these two parts co-operate properly. Configuration of the NPI PIM Read Interface is given in Appendix B.

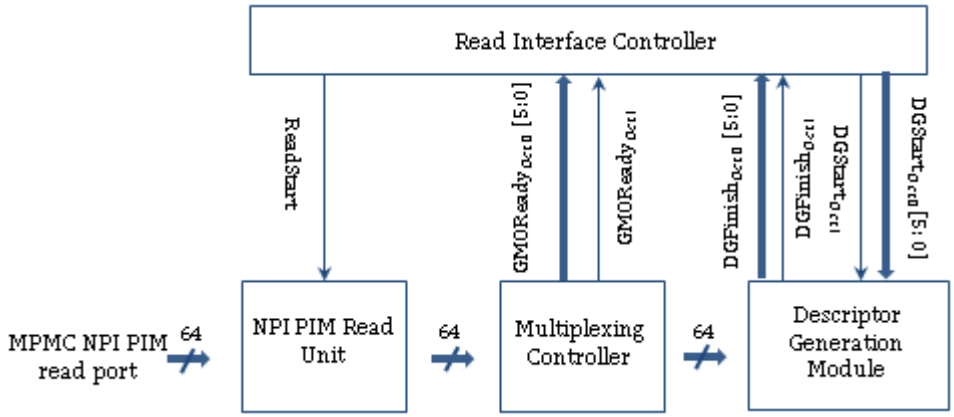


Figure 5-27: Overview of the memory access solution to NPI PIM read interface.

Figure 5-28 shows the pipelined architecture of the partition-based memory access solution. The processing time varies for each partition, which is proportional to the number of keypoints within each partition. However, the time requirement for accessing GMOs from DDR3 is approximately the same since each partition is of the same size.

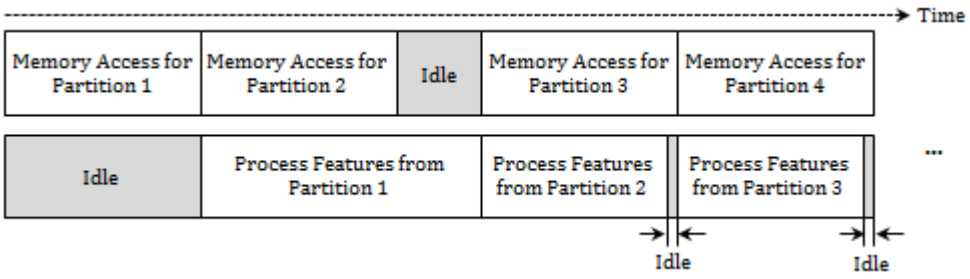
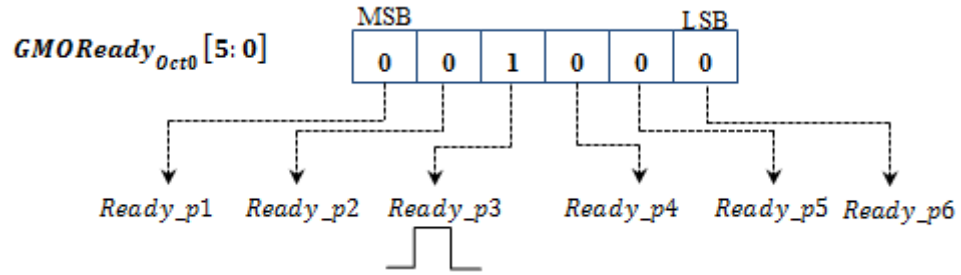


Figure 5-28: Pipelined architecture for the partition-based memory access solution.

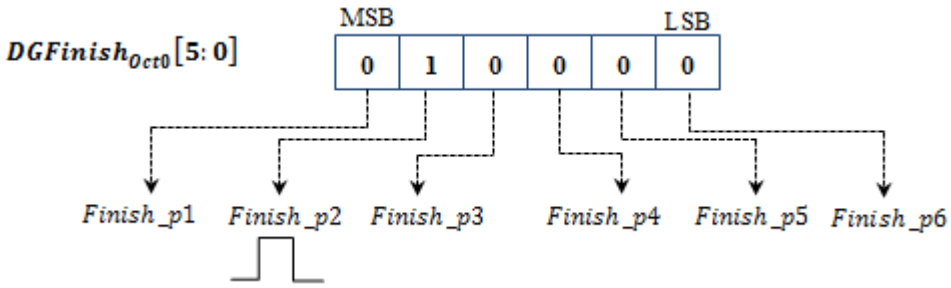
1) Status Indicators

The GMOs from octave 0 are divided into six partitions and octave 1 is taken as one partition. Therefore, the status indicators for octave 0 is a six-bit big endian signal with each bit active high. As shown in Figure 5-29, the Most Significant Bit (MSB)

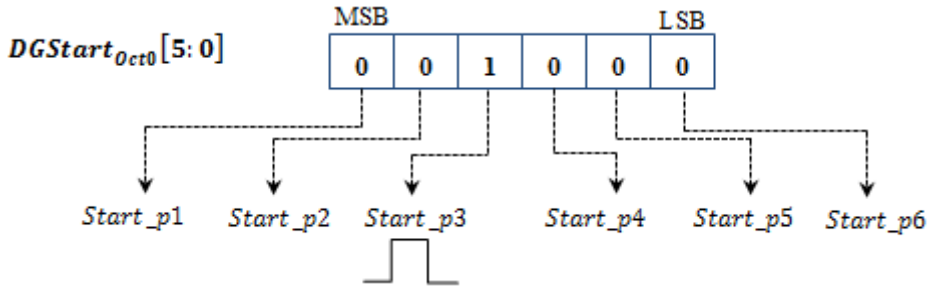
holds the indicating bit for the first partition of octave 0, and the Least Significant Bit (LSB) holds the indicating bit for the last partition of octave 0.



(a)



(b)



(c)

Figure 5-29: An example of the status indicators for octave 0.

2) Read Interface Controller

Detailed flowcharts of the control logic are provided in Figure 5-30 and Figure 5-31. Figure 5-30 shows that DGM does not start processing a newly arrived partition until keypoints from the previous partition have been processed. Figure 5-31 shows that DGM always waits for the corresponding indicating bit ($DGStart_{Oct0}[5:0]$,$DGStart_{Oct1}$)

to be asserted before it starts processing a new partition. Once the current partition has been processed, the corresponding indicating bit ($DGFinish_{Oct0}[5:0]$,$DGFinish_{Oct1}$) is asserted, which indicates that DGM is in the state IDLE and is ready to process next partition. For example, when GMOs from partition 3 is ready in the buffer, the buffer status indicator is set as shown in Figure 5-29(a). The Read Interface Controller waits for the finish indicating bit ($Finish\_p2$) for partition 2 to be asserted, as shown in Figure 5-29(b). Then DGM starts processing partition 3 when the start indicator has been set as shown in Figure 5-29(c).

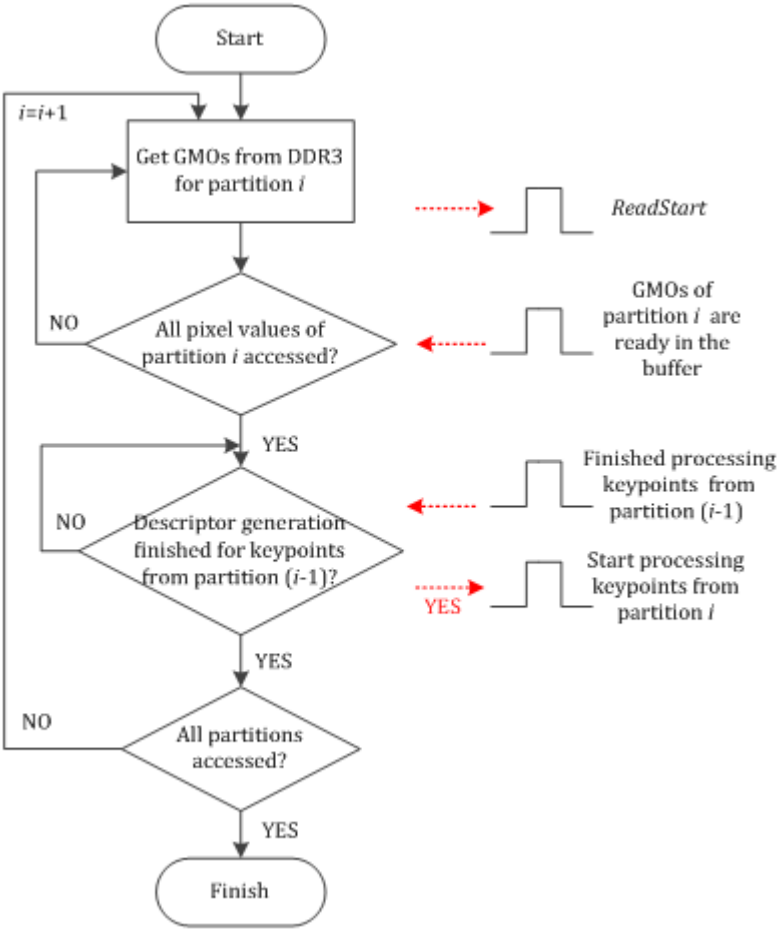


Figure 5-30: General flowchart of the Read Interface Controller.

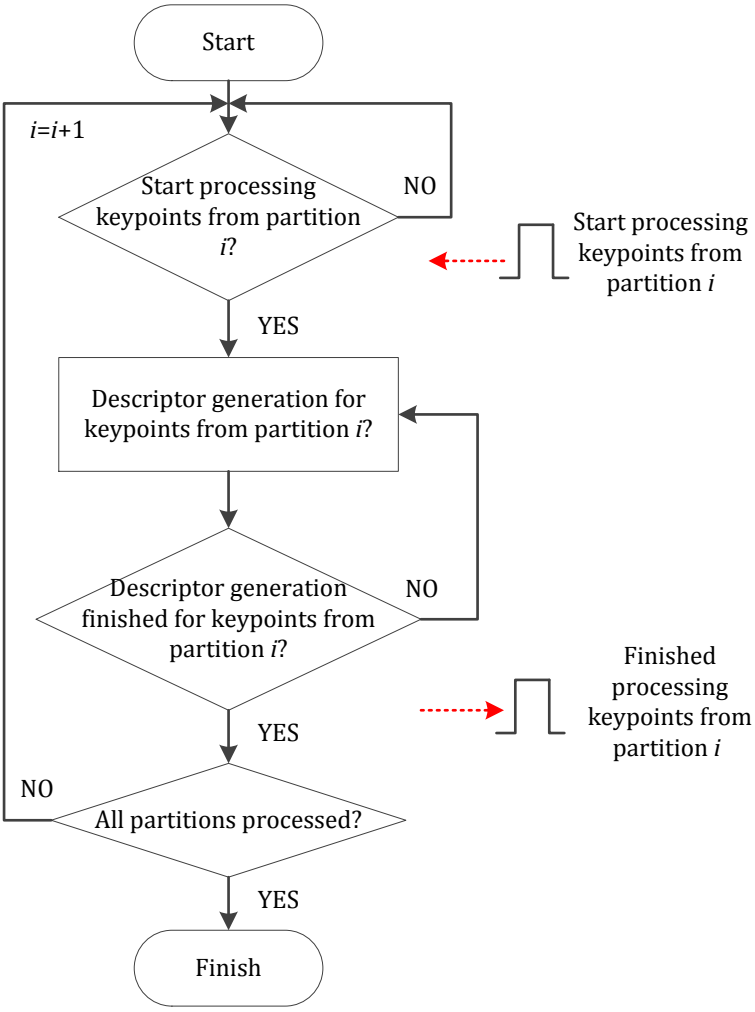


Figure 5-31: General flowchart of DGM.

In a short summary, two conditions should be met before DGM starts processing keypoints from a new partition.

- GMOs of a new partition have been accessed from DDR3 and buffered, ready to use.
- Finish indicating bit has been asserted, indicating that the previous partition has been processed and DGM is in the state IDLE and is ready to process the next partition.

3) Multiplexing Controller

The block diagram of the Multiplexing Controller is shown in Figure 5-32. This unit works with two different clock domains, where data received from DDR3 are first pushed into the FIFO under a clock frequency of 200 MHz, and then sent to ping pong buffers that operate with a clock frequency of 100 MHz.

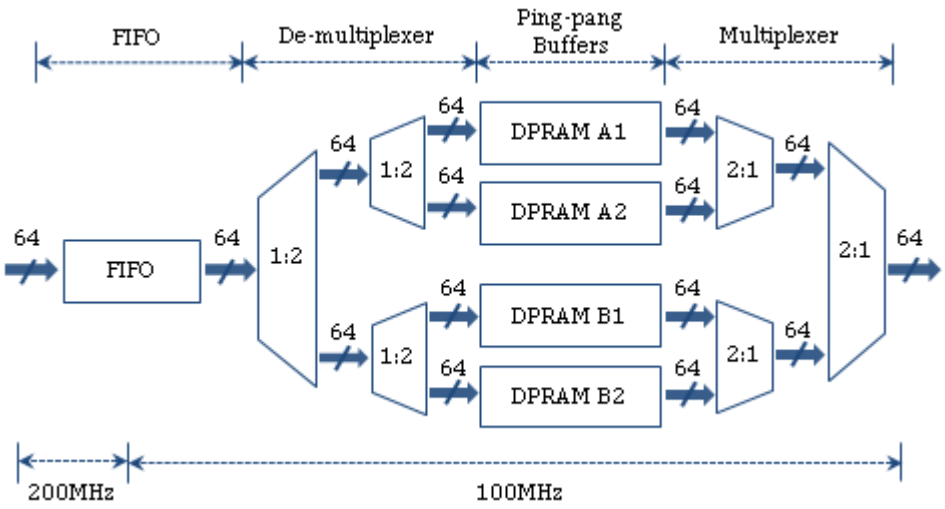


Figure 5-32: Block diagram of the Multiplexing Controller.

This unit mainly consists of four parts:

- FIFO: facilitate data exchange across independent clock domains.
- De-multiplexer: send data to ping pong buffers alternatively for continuous data transfer.
- Ping pong buffers: two groups of buffers with each buffering a partition of GMOs from one of the two pre-selected scales alternatively. The two groups of buffers work in a way that one group is being written while GMOs in the other group is being transmitted to DGM.
- Multiplexer: route data from ping pong buffers to DGM.

During a system design, there are many components that work with different clock domains. Asynchronous FIFO plays an important role in the exchange of data that

are consecutively transferred across different clock domains. The asynchronous FIFO has two interfaces, one for pushing data into the FIFO and the other for reading the data out. Each interface has its own independent clock signal. For example, as shown in Figure 5-33, System X pushes data into the FIFO on *Clock_X* and System Y reads data out on *Clock_Y*. Signal *fifo_full* and *fifo_empty* are employed to take care of the overflow and underflow conditions, respectively.

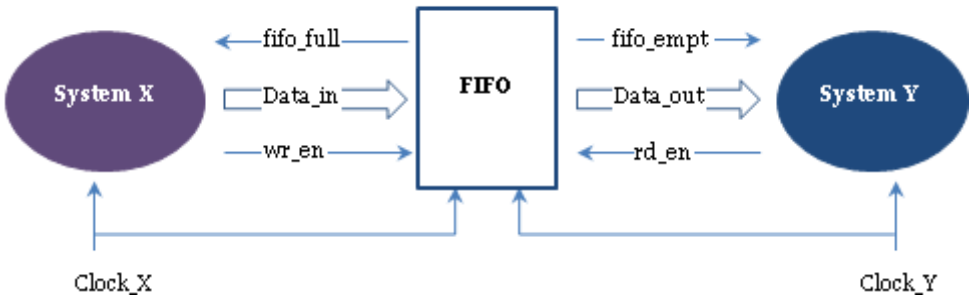


Figure 5-33: Asynchronous FIFO in between two systems with independent clock domains.

b. SRT based Reconfigurable Divider

Figure 5-34 shows the flow chart of the SRT-based divider, with which the division operation is replaced by simpler operations, such as shift, compare, and subtract. The register REG is first initialised with the most significant N bits of the dividend and is compared with the divisor, where N is the word length of the divisor. In the following iteration cycles, the dividend is continuously shifted into REG bit by bit, which is compared with the divisor to decide the corresponding bit of the quotient. The index controls the division process and is initialised to $(M-N)$. After each comparison iteration, index minus by one and the divider finishes when index=0.

Table 5-7 gives an example to the SRT-based divider with the word length of dividend and divisor set to M=12 and N=5, respectively. The REG is initialised with the most significant 5 bits of the dividend (“10100”), and the bit of dividend with underline is continuously shifted into the register REG from the rightmost end.

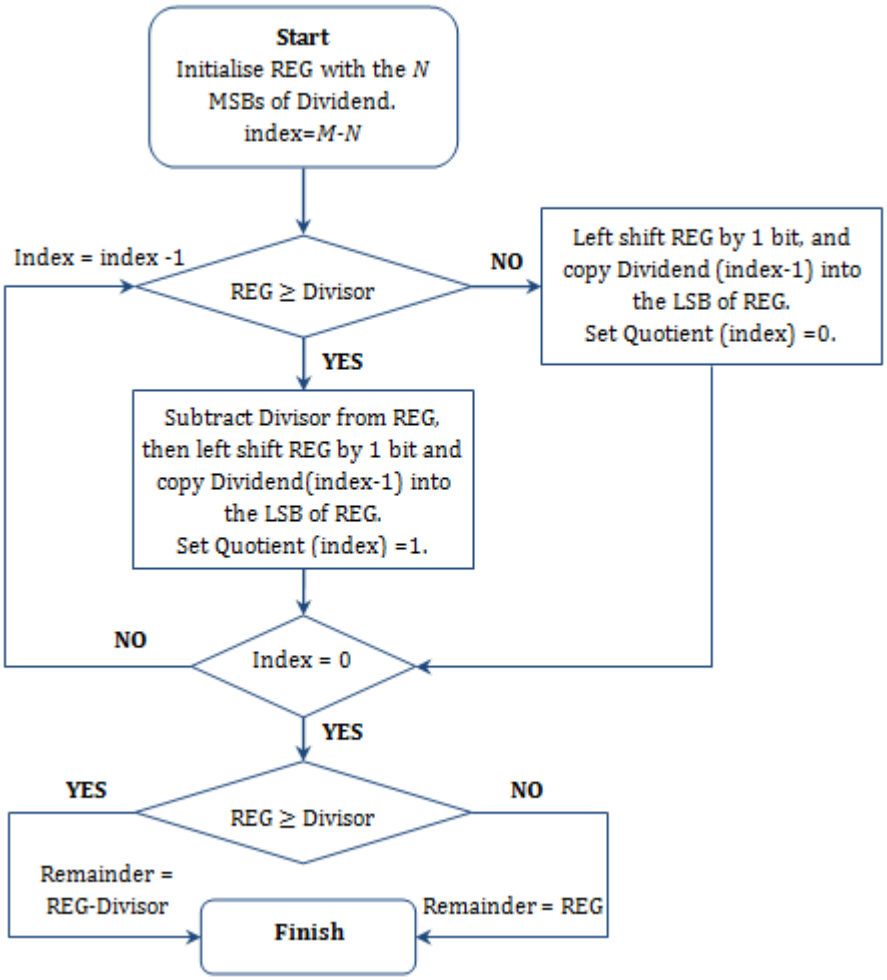


Figure 5-34: Flow chart of the SRT-based divider.

Table 5-7: An example of the SRT-based divider.

Iteration	Index	Dividend (*M*=12) "1010 0000 1101"	Divisor (*N*=5)	REG	Compare (Divisor ≤ REG)	Difference	Quotient	Remainder
Initialize	7	"1010 0000 1101"	"10011"	"10100"	-	-	"00000000"	-
0	7	"1010 0000 1101"		"10100"	YES	"1"	"10000000"	
1	6	"1010 0000 1101"		"10"	NO	-	"10000000"	
2	5	"1010 0000 1101"		"100"	NO	-	"10000000"	
3	4	"1010 0000 1101"		"1000"	NO	-	"10000000"	
4	3	"1010 0000 1101"		"10001"	NO	-	"10000000"	
5	2	"1010 0000 1101"		"100011"	YES	"10000"	"10000100"	
6	1	"1010 0000 1101"		"100000"	YES	"1101"	"10000110"	
7	0	"1010 0000 1101"		"11011"	NO	"1000"	"10000111"	"1000"

The divider can be configured with dividend and divisor ranging from 2 to 35 bits and 1 to 26 bits, respectively. The resource usage of the SRT-based divider is

relatively low when compared with dedicated IP cores provided by Xilinx, as shown in Table 5-8.

Table 5-8: Resource usage comparison of different solutions to divider.

	Radix-2	**High-Radix**	**SRT-based divider**
Target Device	Xilinx Virtex-6 FPGA		
Dividend (bits)	32	37	35
Divisor (bits)	32	24	26
LUTs	2,126	532	336
FFs	3,202	795	165
DSP48E1	0	11	0
RAMB18E1	0	1	0

c. SRT based Reconfigurable Square Root Calculator

Figure 5-35 shows the flow chart of the SRT-based square root calculation, with which the square root computation is replaced by simpler operations, such as shift, compare, and subtract. The register iRight is first initialised with the two most significant bits of the radicand and is compared with iLeft, which is initialised to '1'. In the following iteration cycles, the radicand is continuously shifted into iRight, which is compared with iLeft to decide the corresponding bit of iSquareRoot holding the square root value. The index controls the division process and is initialised to (N-1), where N is the word length of the radicand. After each comparison iteration, index minus by two and the calculation finishes when index=1.

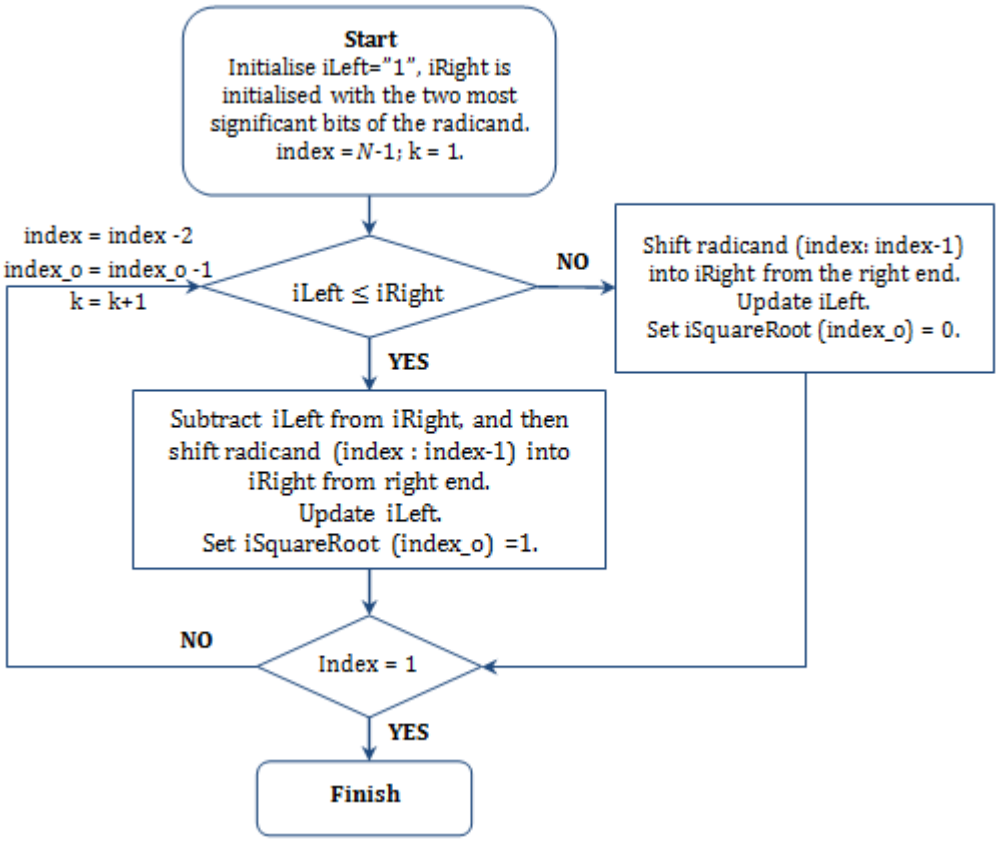


Figure 5-35: Flow chart for reconfigurable square root calculation.

Register iLeft is updated following the flowchart shown in Figure 5-36, where k is the index to the current iteration cycle.

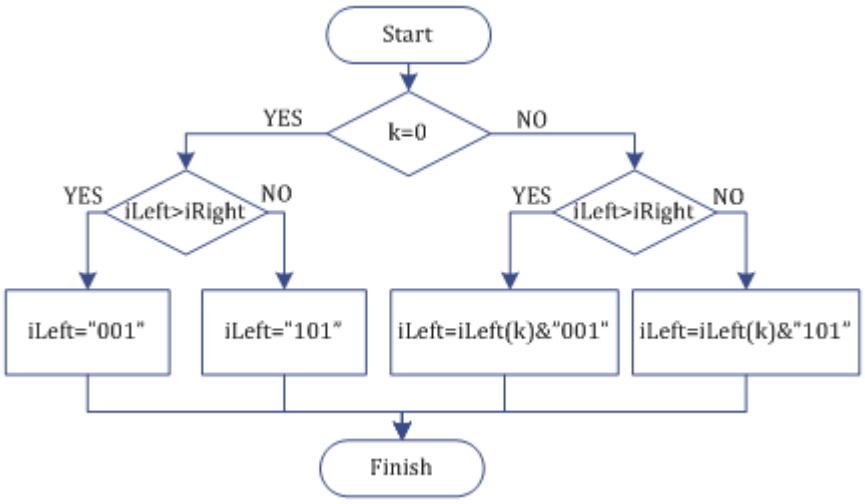


Figure 5-36: Update register iLeft.

Table 5-9 gives an example to further explain the reconfigurable square root calculator. The two bits with underline are the bits to be shifted into iRight from the rightmost end.

Table 5-9: An example of the reconfigurable SRT-based square root calculator.

Iteration (k)	Index	Radicand (N=12) 1010 0000 0000	iLeft	iRight	Compare (iLeft ≤iRight)	Difference	iSquareOut	Index_o
Initialize	11	1010 0000 0000	"1"	"10"	-	-	"000000"	5
1	11	**10**10 0000 0000	"1"	"10"	YES	"1"	"100000"	5
2	9	10**10** 0000 0000	"101"	"110"	YES	"1"	"110000"	4
3	7	1010 **00**00 0000	"1101"	"100"	NO	-	"110000"	3
4	5	1010 00**00** 0000	"11001"	"10000"	NO	-	"110000"	2
5	3	1010 0000 **00**00	"110001"	"1000000"	YES	"1111"	"110010"	1
6	1	1010 0000 0000	"1100101"	"111100"	NO	-	"110010"	0

The reconfigurable square root calculator supports input/output of up to 48bits. Table 5-10 shows that the requirement of the SRT-based method is relatively low when compared with that of the dedicated IP core provided by Xilinx.

Table 5-10: Resource usage comparison of different solutions to square root calculation.

Method	Target Device	Input / Output Width	LUT6-FF pairs	LUTs	FFs	Max Frequency (MHz)
CORDIC	Xilinx Virtex-6 FPGA	48	2,549	2,448	2,511	277
SRT-based			1,696	1,495	1,068	228

5.3.5 Descriptor Matching

This stage maps each keypoint from input images to a corresponding point from the reference image. Descriptor vectors of each keypoints are buffered and the keypoints are matched using the novel matching strategy presented in Chapter 3. This stage is a

task well-suited for hardware implementation, considering that the time consuming descriptor matching process can be compensated by exploring the inherent parallelism of embedded hardware devices. Figure 5-37 shows the block diagram of the descriptor matching module, which consists of one Get Descriptor unit and two identical Compare Descriptor units.

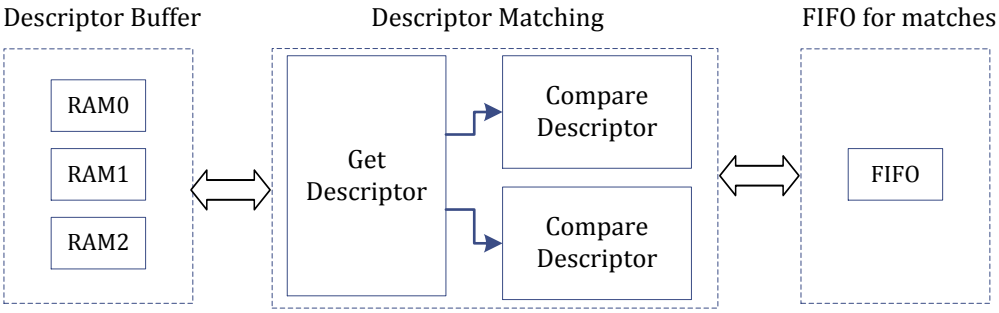


Figure 5-37: Block diagram of the Descriptor Matching module.

Descriptor Buffer: The design can be configured to work in two different modes. In the first mode, each image is compared with its previous frame. In this mode, RAMs are accessed in a way shown in Figure 5-38(a), where one RAM is being written while the other two are being read. In the second mode, the input images are continuously compared with the same reference image. Descriptors from the reference image are buffered in RAM0 and act as the database. Descriptors from consecutive input images are mapped into ping pong buffers RAM1 and RAM2 in turn, with which one is being written by the descriptor generation module, while the other one is being read by the descriptor matching module, as shown in Figure 5-38(b). With DPRAM acting as the buffer, two descriptors arrive every clock cycle, corresponding to a throughput of 0.2G descriptors per second with a clock frequency of 100 MHz. Because DPRAM supports performance of up to 450 MHz, the throughput can be further improved by using higher clock frequency if necessary.

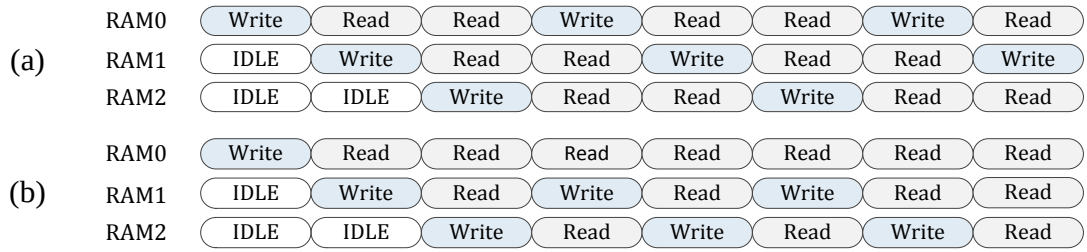


Figure 5-38: Descriptor buffer access.

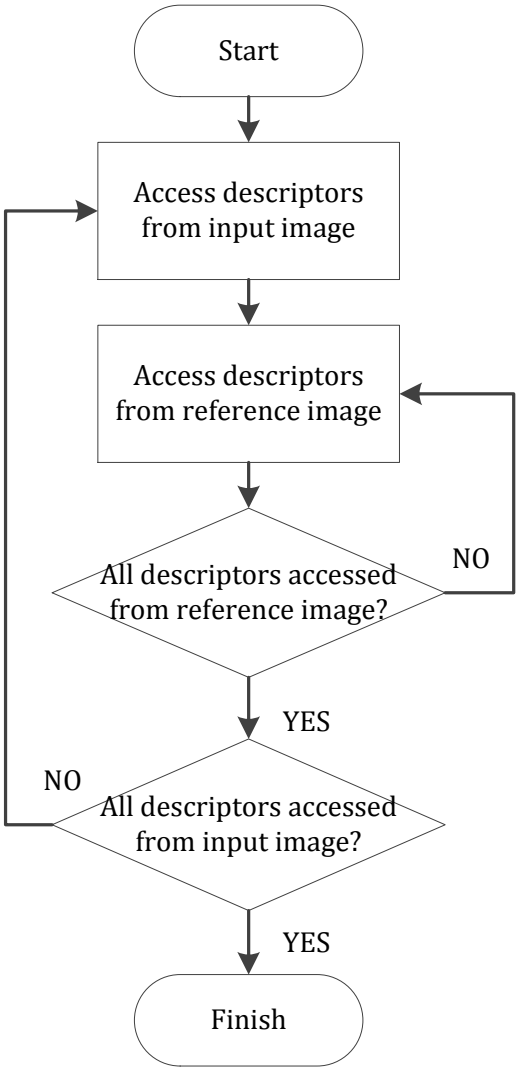


Figure 5-39: Flowchart for *Get Descriptor* unit.

Get Descriptor: This unit is responsible for accessing descriptors from Descriptor Buffer and routing data to Compare Descriptor units. For each descriptor from the input image, all the descriptors from the reference image have to be accessed from the buffer, as shown in Figure 5-39. The corresponding throughput requirement is 0.24G descriptors per second. Because two descriptors are accessed from the buffer every clock cycle, they can be matched against the reference image in parallel. As a result, descriptors from the reference image can be shared by two matching processes and the throughput requirement is reduced to half (0.12G).

Compare Descriptor: This unit compares each descriptor from the input image against the descriptors from the reference database, and mainly consists of the following four steps:

1) Calculate the distance (Δd) between each dimension of the pair of descriptors. This process is iterated n_{ref} times for each descriptor from the input image, where n_{ref} denotes the number of descriptors from the reference database.

2) Count the number ($N_{\Delta d}$) of dimensions with Δd below the pre-defined threshold ($Thr_{\Delta d}$) and keep the two pairs of descriptors with the largest and the second largest $N_{\Delta d}$, which corresponds to the closest and the second-closest neighbour, respectively.

3) Compare $N_{\Delta d}$ of the closest neighbour with a pre-defined threshold.

4) Compare $N_{\Delta d_{second}}$ with $N_{\Delta d_{closest}} \cdot Thr_{\frac{N_{\Delta d_{second}}}{N_{\Delta d_{closest}}}}$.

The above mentioned four steps are iterated n_{inp} times, where n_{inp} denotes the number of descriptors from the input image to be matched against the reference database. By registering intermediate results of each step, the Compare Descriptor unit is able to process descriptors that are continuously received from Get Descriptor unit. The overall timing diagram for the descriptor matching module is shown in Figure 5-40, showing descriptors reading and matching.

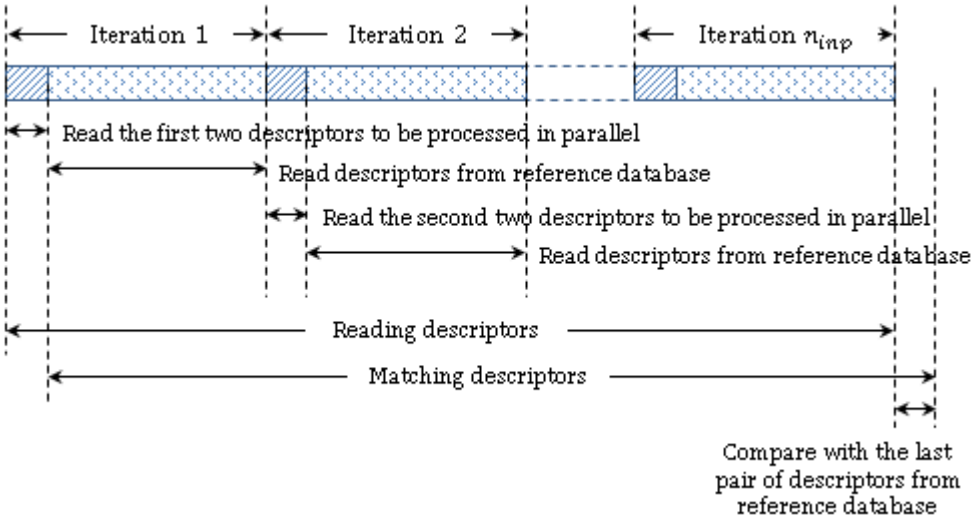


Figure 5-40: Overall timing diagram of the descriptor matching module.

5.4 Trade-off between Resource Usage and System Performance

Resource usage is an important criterion that evaluates the implemented system. In this design, the resource usage of the SIFT processing core falls into two categories: (a) Logic and memory that are required to implement the processing core itself and are independent of the number of pixels processed in parallel, such as the RAM buffering the scaled images and DoG values. These resources increase with the frame size. (b) Memory for storing intermediate calculating results that increases linearly with the parallelism level of the design, such as the number of pixels processed in parallel that requires larger register to hold neighbouring filtered pixels for GMO calculation.

Throughput is an important parameter to assess the efficiency of the processing core, since the high frame rate is the primary target of this project. In this section, throughput refers to that of the SIFT processing core. The throughput of the complete platform with camera front-end and USB back-end will be discussed in Chapter 6. The throughput of the SIFT core can be expressed as the number of frames that the core is able to process per second. Throughput can also be expressed as the number of pixels that are correctly detected, described and matched per second.

It has been mentioned in section 5.3.1 that the SIFT processing core has been arranged into a three-stage pipelined architecture, with which the overall throughput is decided by the stage that consumes the longest time. It can be seen from equation (5.2) that the key factor that affects the throughput of feature detection is the size (k_G) of the largest Gaussian kernel applied, followed by the number of pixels (n_{pixel}) smoothed in parallel. The throughput of feature detection decreases as k_G increases, but increases linearly with the number of pixels processed in parallel, as shown in Table 5-2. Smaller kernel size results in a higher frame rate for feature detection, but potentially increases the processing time of descriptor generation and matching as a result of the larger number of keypoints detected due to over-detection. Besides, with the increase of k_G and n_{pixel}, the number of multiplications increases accordingly and extra resources are required to buffer the intermediate results of the following calculation steps. Therefore, two pixels are filtered in parallel using Gaussian kernel of k_G=31, with which the design achieves at least 60 fps while keeping relatively high accuracy and low resource usage. The relationship between the throughput, accuracy and resource usage of the feature detection module is shown in Table 5-11 and Table 5-12.

Table 5-11: Relationship between throughput, accuracy and resource usage with respect to Gaussian kernel size k_G for feature detection module.

Gaussian kernel size k_G	Throughput	Accuracy	Resource Usage
↑	↓	↓	↓

Table 5-12: Relationship between throughput, accuracy and resource usage with respect to the parallelism level of feature detection.

Parallelism level	Throughput	Resource Usage
↑	↓	↓

Section 5.3.4 has shown examples of the trade-off between the accuracy, memory and throughput. By taking advantage of the LUT-based processing method, it takes only four clock cycles to calculate one pixel orientation with relatively high accuracy. By representing each GMO with 16 bits, four GMOs are concatenated as one data to make full use of the MPMC interface, which reduces the throughput requirement of the MPMC interface. Besides, on-chip memory requirement for the partition-based memory access solution is also reduced, which is proportional to the word length of GMOs. Therefore, the slight degradation in the accuracy of GMOs leads to the decreased requirement in both the on-chip memory and the throughput of MPMC interface.

5.5 Summary

This chapter presents the hardware architecture of the SIFT processing core with all phases of the algorithm covered, including feature detection, descriptor generation and descriptor matching. With the pipelined and parallel structure developed, the SIFT processing core is fully embedded on-chip and is able to process VGA images at least 60 fps with a system clock of 100 MHz.

In feature detection module, pixels can be constantly streamed into the processing core and filtered with relatively low computation cost as a result of the SRT-based pixel streaming strategy. Efficient memory solutions have been proposed for Gaussian smoothed images and DoG values. The memory requirement remains constant with the increase of the parallelism level of the SIFT processing core, which is one of the contributions of this work. Besides, GMOs are buffered in on-board DDR3, which offers 512 Mbytes memory. Each GMO is represented by 16 bits, which saves the on-chip memory requirement of the partition based memory solution and reduces the throughput requirement of the MPMC interface while preserving relatively high accuracy. The throughput of feature detection can be increased by increasing the parallelism level of the design at the expense of a small amount of resources for buffering intermediate results, such as the SRT holding pixels for Gaussian filter and that holding neighbouring filtered pixels for GMO calculation, etc. The throughput can also be increased by using Gaussian kernel of smaller sizes,

which increases the throughput at the expense of accuracy and scale invariance and is not an optimal choice.

In descriptor generation module, each feature point takes only 7.57 us to be generated as a result of the polar sampled spatial arrangement of SRI-DAISY, the LUT-based Gaussian smooth and interpolation, and the SRT-based square root computation and division. The design processes up to 132,100 descriptors per second at a system frequency of 100 MHz, which is fast enough to generate descriptor for VGA resolution video of at least 60 fps, provided that there are no more than 2,200 keypoints per frame.

The descriptor matching module implemented the novel matching strategy, which achieves a throughput of 0.2G descriptors per second with a clock frequency of 100 MHz. Because this module does not include complex computations, such as square root computation, the resource usage is low and the throughput can be increased by running several modules in parallel.

The SIFT processing core is incorporated into a platform with a camera front-end and a USB back-end, which will be introduced in Chapter 6.

Chapter 6 An Image Matching System based on the Optimised SIFT

6.1 Introduction

Design parameters have been evaluated in Chapter 4 to show that high performance and high accuracy can be achieved by the hardware design for the optimised SIFT algorithm. The hardware architecture of the SIFT processing core has been presented in Chapter 5, which can be integrated into an FPGA device. In this chapter, an image matching system is described in which the SIFT processing core presented in Chapter 5 is integrated into an embedded system that communicates with a camera front-end and a USB back-end, as shown in Figure 6-1. Besides, three types of experiments are conducted to verify the system performance. Hardware efficiency of the design is compared with existing solutions in this chapter.

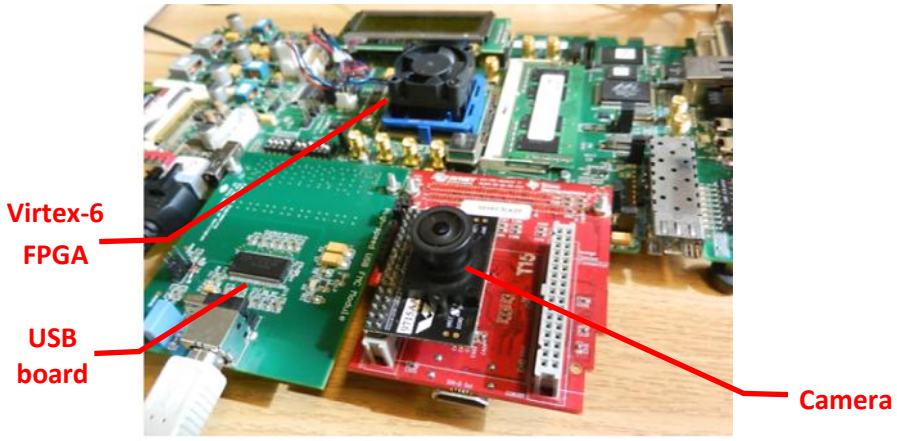


Figure 6-1: The SIFT based image matching system.

6.2 Embedded System in FPGA

As mentioned in Chapter 5, the contemporary FPGA devices are rich in resources and FPGA vendors support a wide range of embedded processing peripheral IP cores so that extensive logic functionality can be designed to work in a single FPGA device. Taking advantage of the reconfigurable property of the FPGA, it is fast and convenient to build a complete digital system on a FPGA device, including processing, controlling and interfacing block, which is a system-on-chip (SoC)

design. An embedded system is developed to operate the SIFT processing core and the entire platform is verified on the Xilinx ML605 FPGA board.

Figure 6-2 shows the block diagram of the SIFT-based image matching system, which shows the main interfacing, processing, controlling and buffering units. Detailed introduction to the camera controller and the USB controller are given in Appendix D.

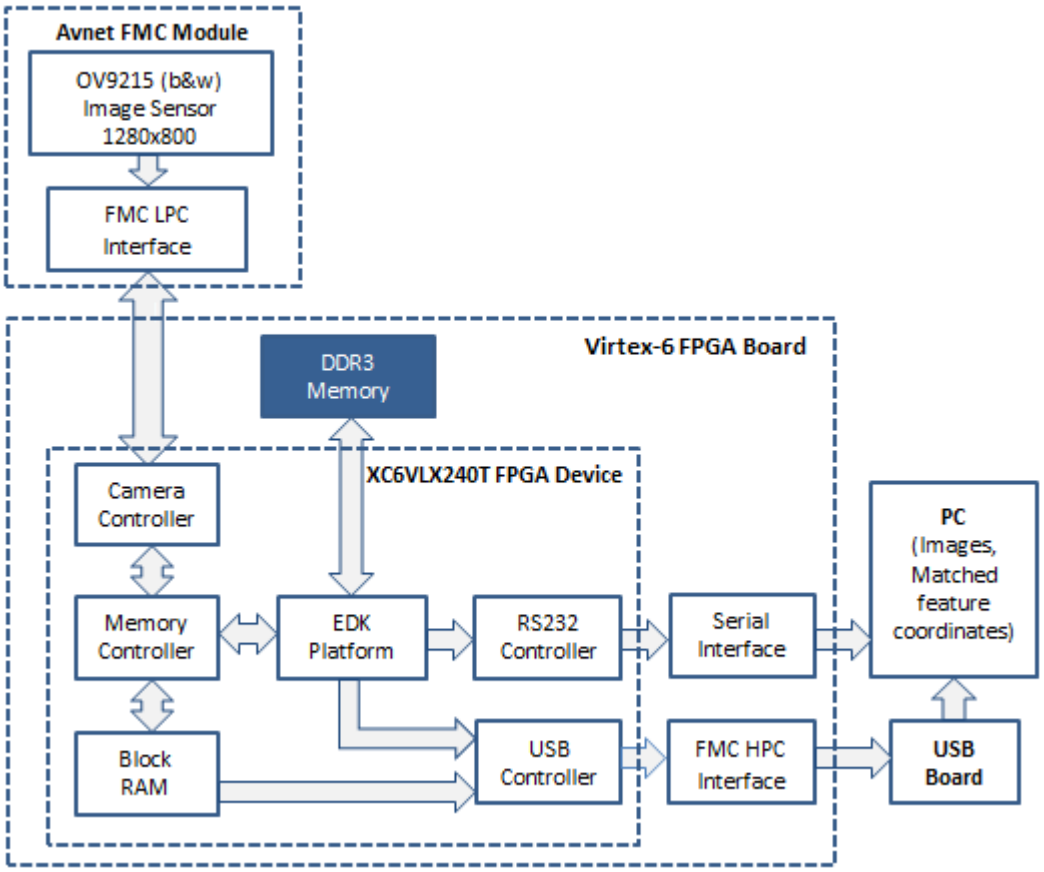


Figure 6-2: Block diagram of the image matching system showing the main interfacing, processing, controlling and buffering units.

The OV9715 image sensor mounted on the Avnet FMC Module is connected to the FPGA board via FMC LPC [70] connector and is configured to deliver 640x480 resolution video at 30 fps. All the control and processing blocks are fully embedded in the FPGA device. The input images and matched keypoints coordinates are

outputted to PC via USB that is connected to the FPGA board through FMC HPC connector. Status of the FPGA system can be monitored by the messages sent to PC via RS232 serial interface. Configuration of the Avnet FMC module with OV9715 OmniVision image sensor is given in Appendix E.

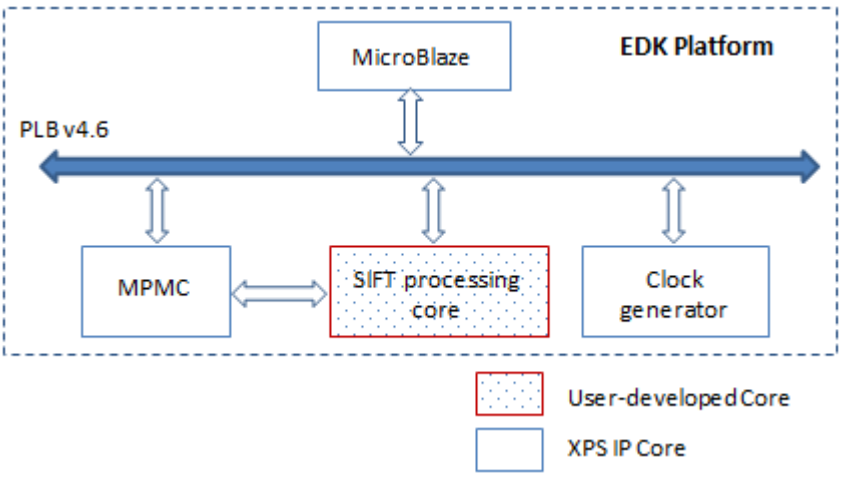


Figure 6-3: Block diagram of the EDK platform.

Figure 6-3 shows the block diagram of the embedded system that is developed by using XPS provided by Xilinx Embedded Development Kit (EDK) [71]. EDK is an integrated development environment for designing embedded processing systems. In this design, the IP cores implemented in the EDK system fall into two categories: XPS pre-built IP cores, such as the MPMC interface for DDR3 access, and the user-developed components, such as the SIFT processing core.

6.3 Experimental Results

In this section, three types of experiments are conducted to verify the system performance. The first experiment examines the system's robustness in presence of different geometric and photometric transformations, such as in-plane rotation and image scaling, changes in viewpoint angle, blur, illumination and noise. This type of experiment is ideal since it does not contain complex cluttered background or partial

occlusion, so the system is integrated into an object recognition application in the second experiment. In the last experiment, matching results from the system are used for video stabilisation, which tests the detection and matching accuracy of the system. It should be noticed that the matching results displayed in the first two experiments are the outputs of the proposed system directly. In the last experiment, RANSAC is applied to eliminate outliers such that the estimation of transformation matrix is more accurate. All experiments are conducted on real world images of size 640x480 pixels.

6.3.1 Experiments using Real World Images

The system performance is tested in presence of different transformations, such as changes in scale and rotation, viewpoint angle, image blur, illumination, and in presence of noise. A reference image, which is photographed on a white background, is matched against itself, but with various transformations. Each solid line connects a pair of keypoints matched using the novel matching strategy presented in Chapter 3.

a. Rotation Invariance

Figure 6-4 shows the matching results for a set of images with rotation of -180 to 180 degrees. The average precision is above 95%, indicating that the SRI-DAISY based matching system is fully invariant to rotation.

Figure 6-4: Testing results for image rotation.

b. Scale Invariance

The matching performance is tested at a variety of scales. Figure 6-5 shows the matching results, where both zoom in and zoom out have been tested. In general, a larger number of keypoints are matched when keypoints from the down-scaled images are matched against the reference, because scales for all keypoints from the down-scaled images would be present in the reference image. It is more challenging to match the images that are scaled up against the reference image, such as the images shown on the last row of Figure 6-5, because the correspondences of many keypoints detected from the up-scaled images are not detected from the reference image. It can be seen from Figure 6-5 that although the number of matches is limited

for images that are scaled by a factor of 0.3 and 3.5, respectively, all of the matches are correct. The average precision is greater than 95%, indicating that the design is robust to scale changes.

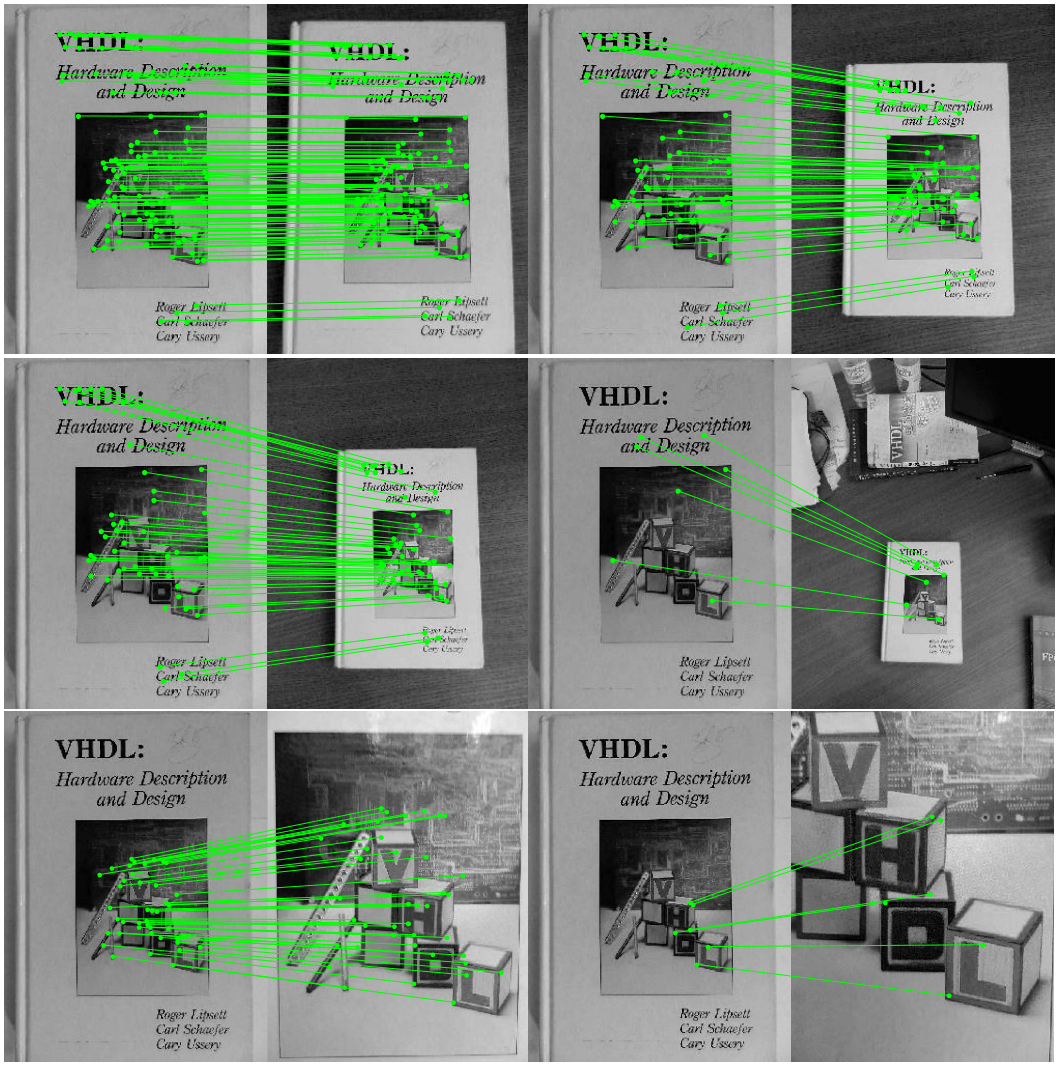


Figure 6-5: Testing results for scale changes. The scaling factors are 0.9, 0.7, 0.6, 0.3, 2 and 3.5, respectively, starting from the upper-left corner.

c. Viewpoint Changes

The robustness is also tested under various projective transformations. The number of matches drops with the increase of viewpoint angle and is reduced to three in a most challenging situation with viewpoint angle of approximately 60 degrees, as

shown in the image on the bottom-right corner of Figure 6-6. In general, the viewpoint changes have a larger impact on the number of matches than the correctness of matches. For viewpoint changes of within 60 degrees, the precision is 85% in the worst case, which is mainly due to the limited number of matches. Therefore, the design is partially robust to viewpoint changes.

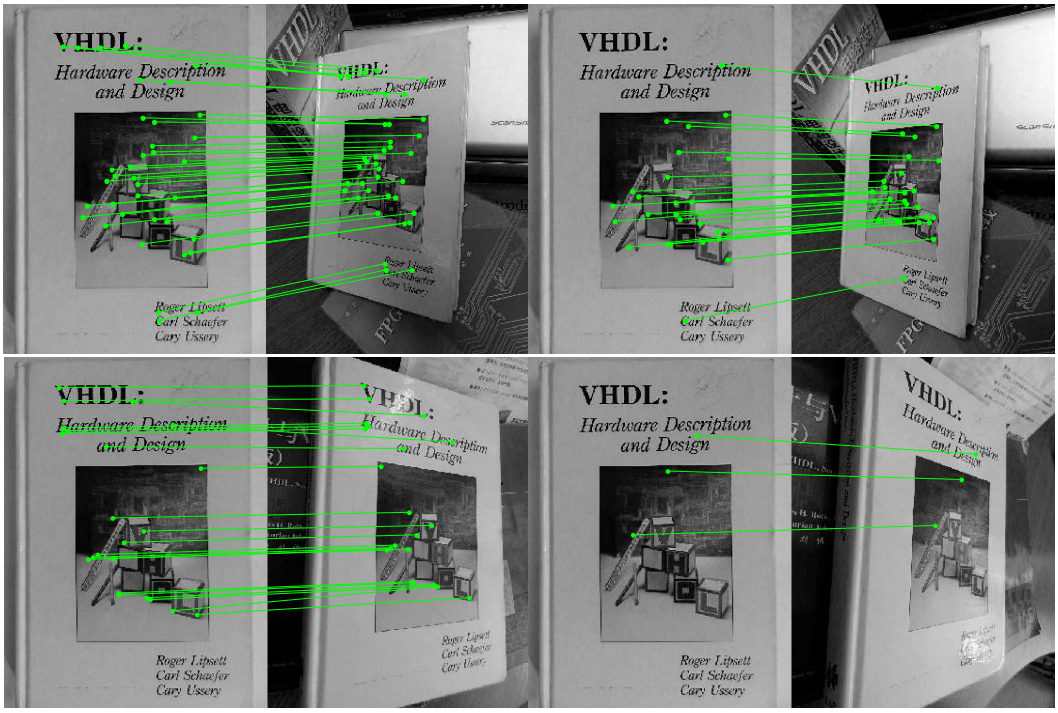


Figure 6-6: Testing results for viewpoint changes.

d. Blur

In presence of image blur, the pixel intensities and shape of local structures change in an unpredictable way. The SIFT descriptor is not designed invariant to such transformation. In this experiment, the camera vibration is created manually in different directions during shooting to produce blurred images. Experimental results are presented in Figure 6-7, which shows that the number of matches decreases with the increasing amount of blur. However, there are still some correct matches in presence of a significant amount of blur, as shown in the image on the bottom-right corner of Figure 6-7.

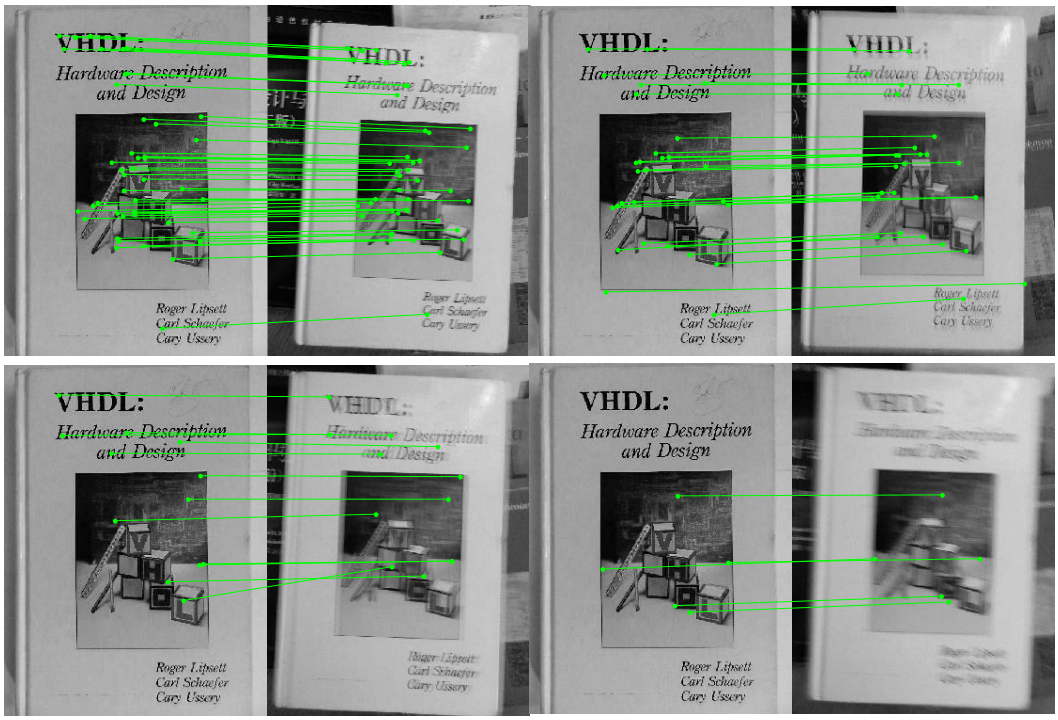


Figure 6-7: Testing results for image blur.

e. Illumination Changes

The illumination invariance is demonstrated in Figure 6-8. The two images are of the same scene from the same viewpoint, except for the difference in illumination. In the first test shown in the left image of Figure 6-8, there are 108 matches in total, 3 of which are incorrect. In a more challenging situation where there exists a significant change in illumination, only 1 of the 60 matches is incorrect, as shown in the right image of Figure 6-8. The reduction of matches is mainly because the correspondences of many keypoints detected from the reference image are not detected from the low contrast (dark) area from the input image.

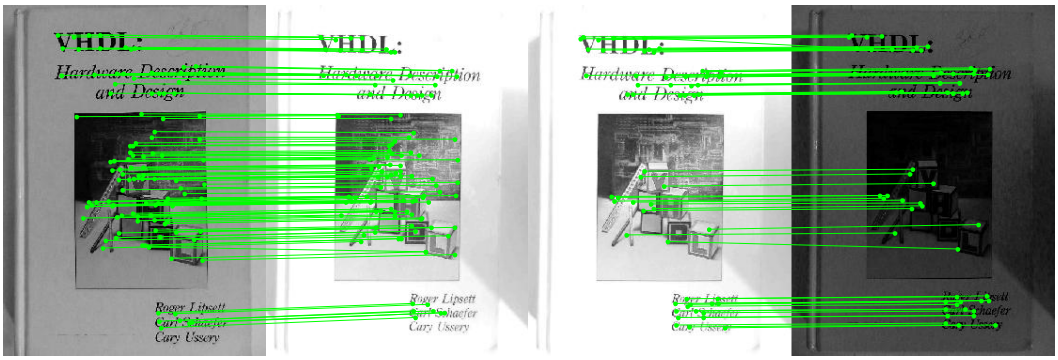


Figure 6-8: Testing results for illumination changes.

f. Noise

The robust to noise is tested by adding up to 3% Gaussian noise to images. A random number from the uniform interval [-7.65, 7.65] is added to each pixel, where the pixel values are in range [0, 255]. In Figure 6-9, the left and the right image show the results with 1% and 3% Gaussian noise added, respectively. The average error is below 5%, and hence the system is robust in presence of noise.

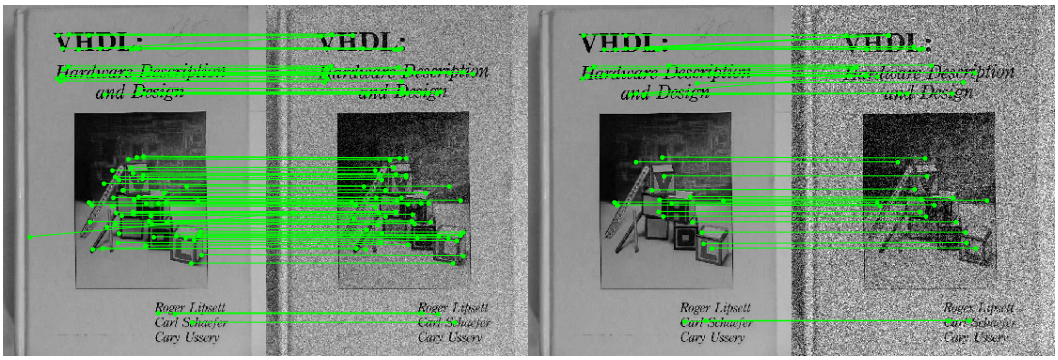


Figure 6-9: Testing results with Gaussian noise added.

g. 3D rotation

The system performance is tested on 3D objects, because SIFT has been widely applied as the first stage of applications, such as 3D reconstruction. The keypoints are detected and matched from adjacent images.

Figure 6-10 shows the matching results for a 3D object that is rotated along x, y and z axis, respectively. Although SIFT is not designed to be invariant to rotation of a 3D object, the system still shows some level of robustness. 3D in-depth rotation along the z-axis is most challenging, and the system is robust to in-depth rotation of up to 20 degrees.

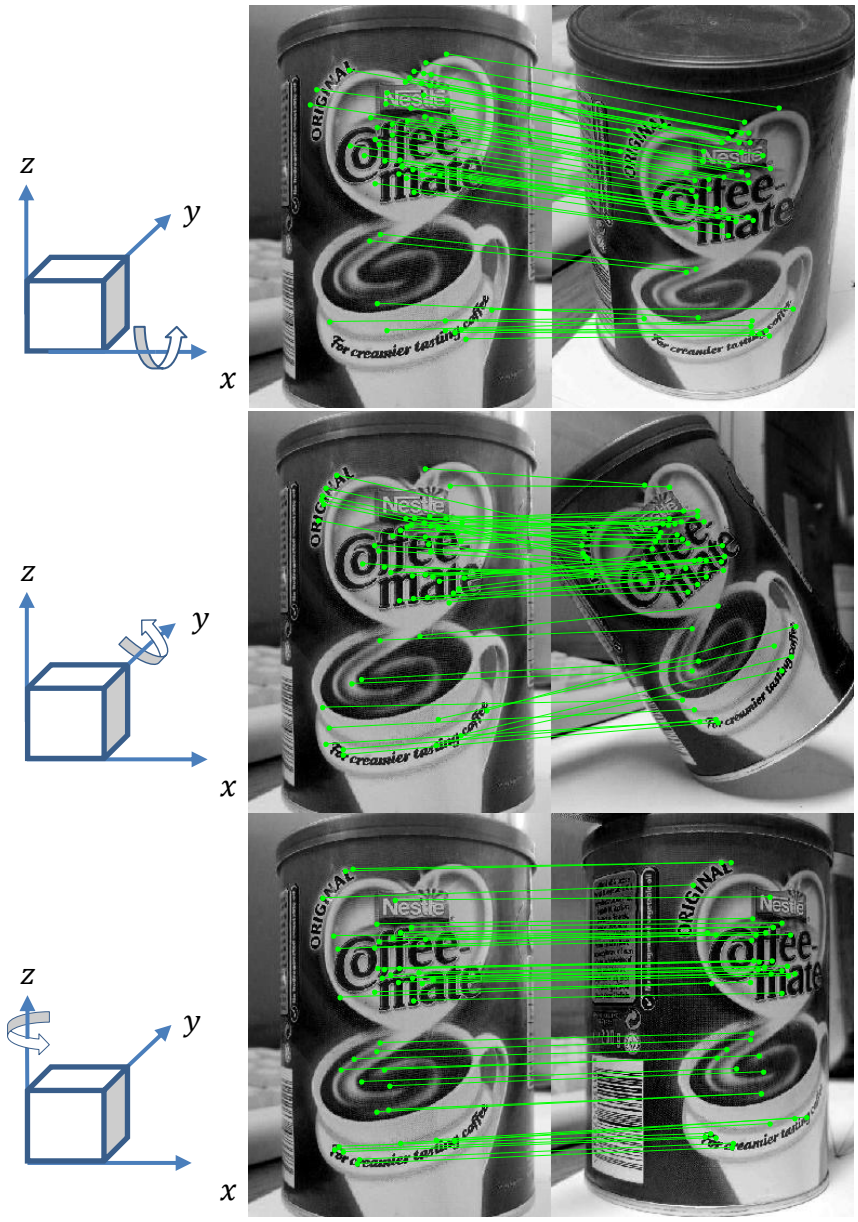


Figure 6-10: Matching results for 3D object that is rotated along x, y and z-axis, respectively.

6.3.2 Application for Object Recognition

The system is tested on a practical application, aiming at object recognition in a typical lab environment. In this experiment, object recognition is formulated as follows: Given a reference image of the target object or scene, keypoints are firstly extracted and each keypoint is associated with a descriptor. Then descriptors are generated for all keypoints that are detected from the input images, whether or not the keypoints are from the target object or scene. Each descriptor in the input images is compared with all descriptors in the reference image. The coordinates of the matched keypoints are buffered. The consecutive input images act as the database containing the target object to be recognised from the reference image. Experiments are conducted on the recognition of both planar and 3D objects. Figure 6-11 shows the flowchart illustrating the work mode for object recognition.

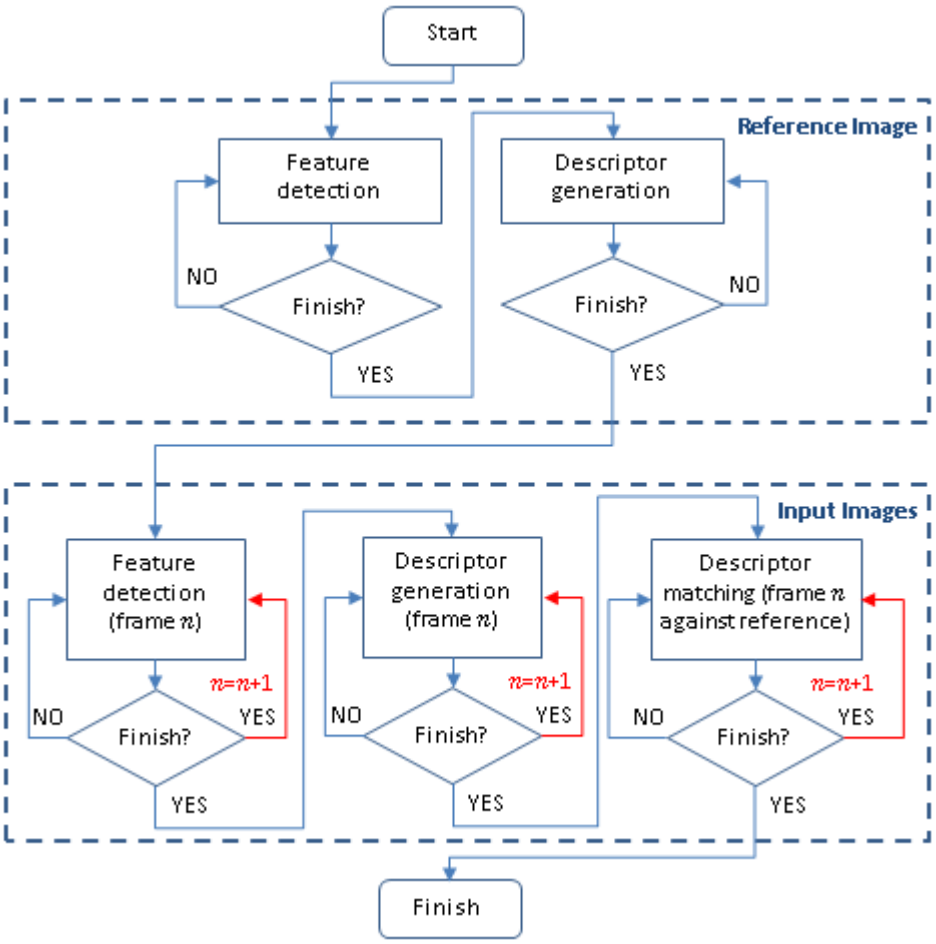


Figure 6-11: Flowchart for the system when used for object recognition.

a. Planar Object Recognition

In this section, the SIFT features are used for object recognition of planar objects. Because object recognition in real world requires objects to be correctly identified in presence of nearby clutter or partial occlusion, experiments are conducted for object recognition from some challenging scenes, where there is a combination of significant amount of transformations, cluttered background and partial occlusion.

Figure 6-12: Object recognition for planar objects in presence of transformations, cluttered background and partial occlusion.

It has been mentioned by Lowe that any three of the correct matches would be sufficient for reliable recognition. The matching results shown in Figure 6-12 are the outputs from the system without outlier elimination, which shows that the system provides high precision matches that can be applied for reliable object recognition.

b. 3D Object Recognition

Figure 6-13: Object recognition for 3D objects in presence of transformations, cluttered background and partial occlusion.

In this section, the SIFT features are used for object recognition of 3D objects, which is more challenging than recognition of planar objects as a result of the lighting condition that is uncontrolled in the lab environment. This will degrade the recognition effect since the illumination changes affect the 3D surfaces with differing orientations by different amounts, which can cause a large change in the relative magnitudes of some gradients. However, it can be seen from the experimental results shown in Figure 6-13 that the system is capable of providing correct matches for the 3D objects recognition.

6.3.3 Application for Video Stabilisation

The previous application concerns more about the quantity of matches than the quality. In this section, experiments are conducted to test the quality of matching. The system performance is tested in the application of video stabilisation, which is used to eliminate unwanted shakiness in the video caused by high frequency movement of the camera while recording. In this experiment, video stabilisation is formulated as follows: Given a video sequence, each keypoint detected from consecutive frames is associated with a descriptor. Each descriptor in the current frame is compared against all descriptors in the previous frame. Then the affine transformation matrix is computed on the matches over successive frames with outliers eliminated by RANSAC, where the outliers correspond to incorrect matches that do not agree with the transformation parameters between images. Transformation matrix is then estimated that represents the inter-frame motion between successive frames. In this experiment, the system contributes to camera motion estimation by detecting salient features that can be tracked over multiple frames of video and matching descriptors between adjacent frames in the video sequence. Figure 6-14 shows the flowchart for video stabilisation.

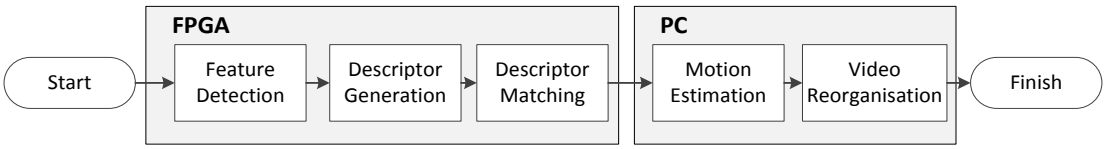
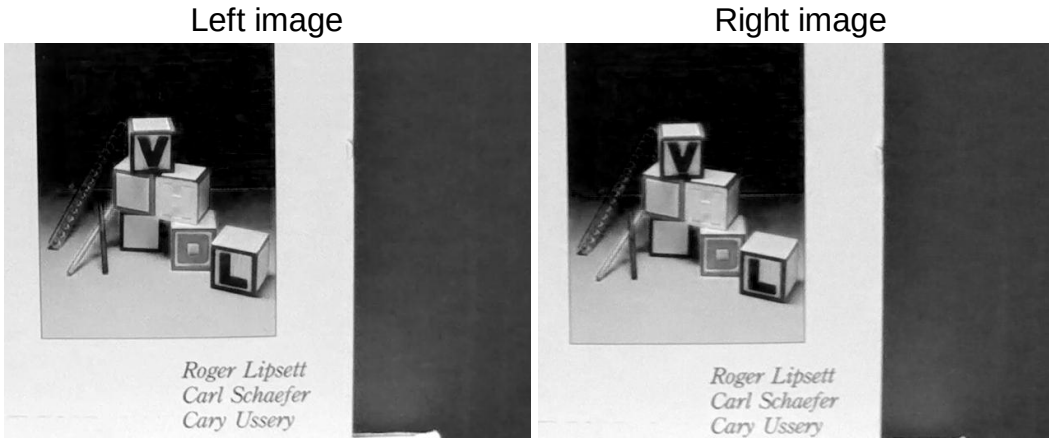
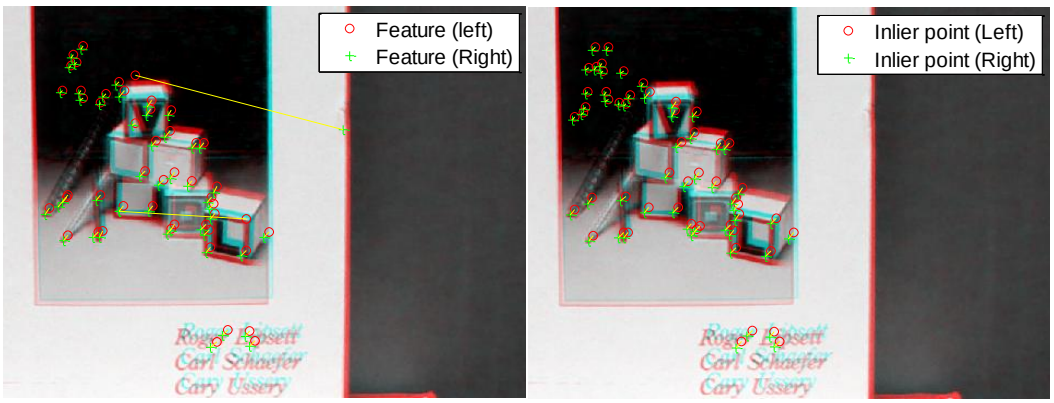


Figure 6-14: Flowchart of the system used as feature tracker for video stabilisation.

It should be noticed that the video sequence used in this experiment is real jittered ones recorded using a hand-held camera for a static view. Actually, a keypoint shift in position can occur not only due to camera shakiness, but also in presence of intentional panning movement or because it belongs to a moving object in the scene, which will result in inaccuracy in motion estimation. Because the aim is to test the detection and matching accuracy of the system, there is no intentional camera movement or moving object in this experiment. Therefore, the misalignment between successive frames is the result of unintentional high-frequency motion, named jitter.



(a)



(b)

Figure 6-15: (a) Two successive frames from a video sequence. (b) Left: original matches; Right: matches after model fitting using RANSAC.

The red-cyan colour composite is used to illustrate the pixel-wise difference between images. It can be seen from Figure 6-15(b) that there is an obvious offset in both vertical and horizontal directions between the two frames shown in Figure 6-15(a). The matched keypoints from the two successive frames are represented by red circle and green cross, respectively, which are connected by yellow lines to show the correspondences selected by using the novel matching strategy.

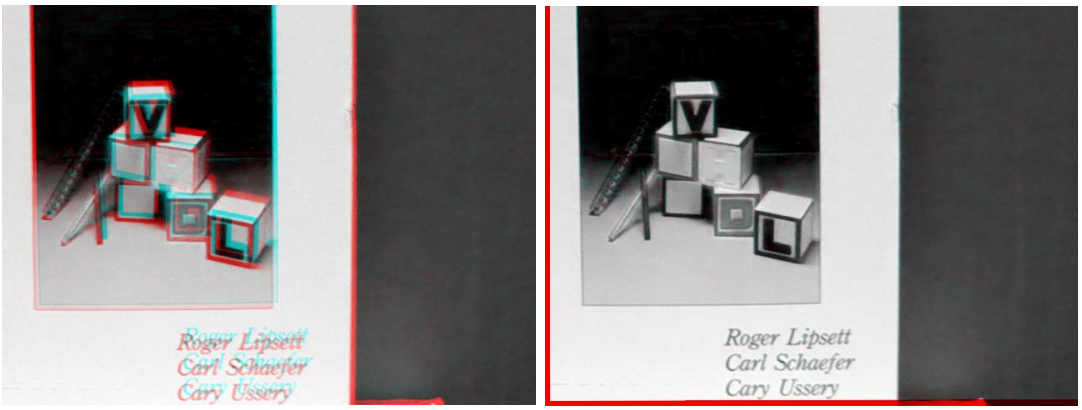


Figure 6-16: Left: Overlay of the original second image and the first frame; Right: Overlay of stabilised second image and the first frame.

The second frame is warped and compared with the first frame. It can be seen from the right image of Figure 6-16 that the original first frame (in red) is well aligned with the stabilised second frame (in cyan), such that the red-cyan composite shown in the left image of Figure 6-16 becomes almost purely black-and-white in the overlapped region, indicating that the pixel-difference between the original first frame and the stabilised second frame is negligible.

Peak Signal-to-Noise Ratio (PSNR) reflects the misalignment between two frames and can be used as a measurement for evaluating the similarity between frames. In this section, it is used to numerically evaluate the stabilisation performance, which is defined as:

$$PSNR(I_a, I_b) = 10 \log_{10} \frac{I_{MAX}^2}{MSE(I_a, I_b)} \tag{6.1}$$

where I_{MAX} is the maximum intensity value of a pixel and is equal to 255. $MSE(I_a, I_b)$ is the mean square error between frame a and frame b, as defined in Equation (6.2).

$$MSE(I_a, I_b) = \frac{1}{w \times h} \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} [I_a(i,j) - I_b(i,j)]^2 \tag{6.2}$$

where w and h are the width and height of input frames, respectively.

There are two PSNR-based evaluation criteria [72] for video stabilisation: Inter-frame Transformation Fidelity (ITF) and Global Transformation Fidelity (GTF). ITF measures the short-time stabilisation between successive frames and shows how good the estimated transformation fits the true transformation. GTF is a long-time measurement that evaluates the motion compensation of the current stabilised frame with respect to the initial reference image. In general, stabilised video should be more continuous than the original sequence, so PSNR should increase from the input sequence to the stabilised one, and hence stabilised sequence should have a higher IFT and GTF than the original input sequence.

Figure 6-17 and Figure 6-18 shows the ITF and GTF for both the original and the stabilised video sequence. In both cases, the curve that represents the stabilised video sequence is always above the original one. Both ITF and GTF of the original video sequence drop as a result of the less overlapping area with the reference frame. Despite of the accumulative error passed down consecutive frames, the high values of PSNR of GTF shows that the fidelity of the system is high. It should be noticed that the fidelity measurement used to evaluate the performance is more indicative than quantitative, because the values depend on the video sequence under consideration.

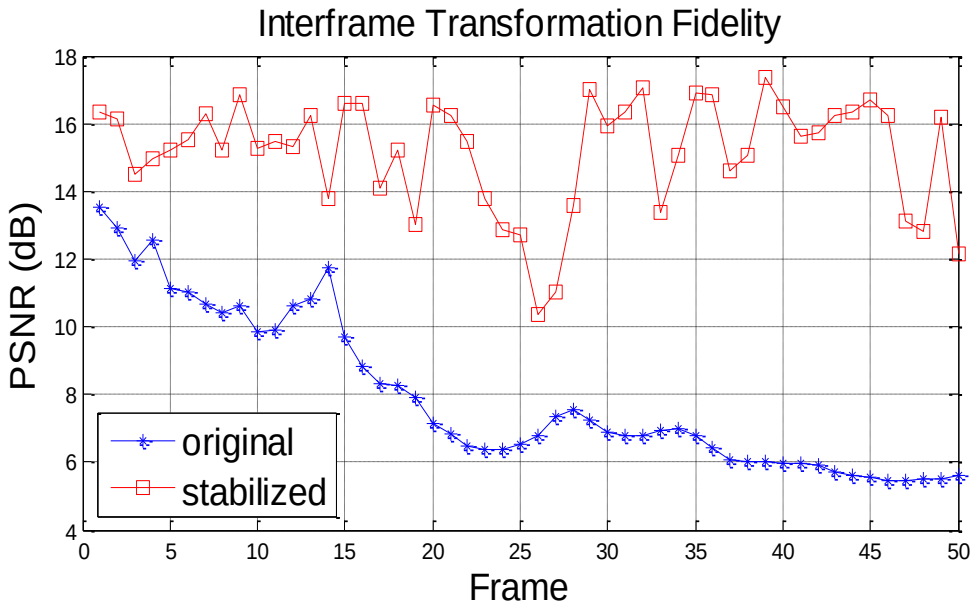


Figure 6-17: PSNR between successive frames for both the original and the stabilised video sequence.

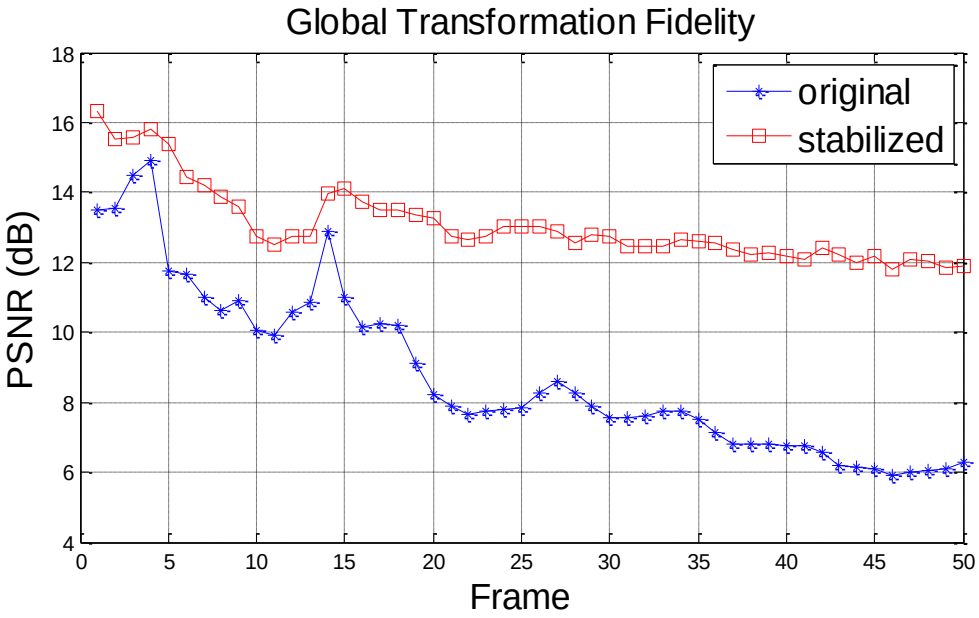
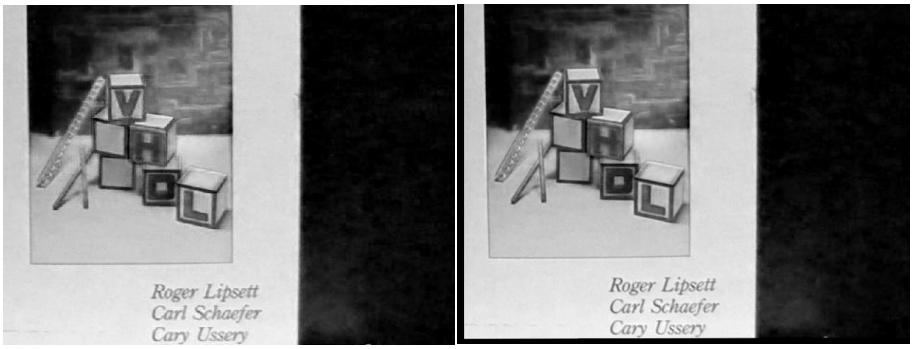
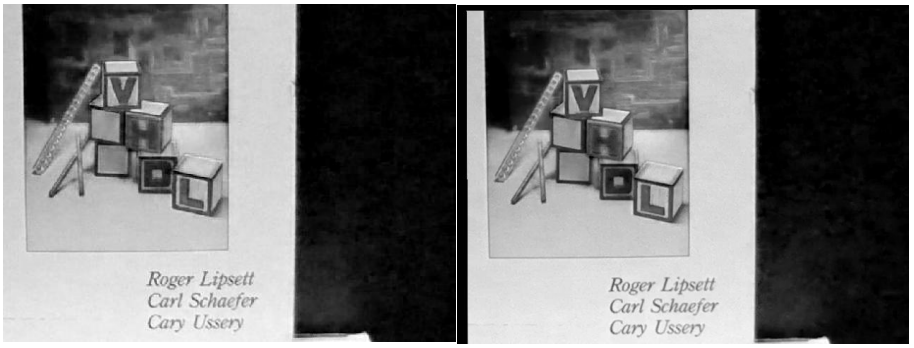


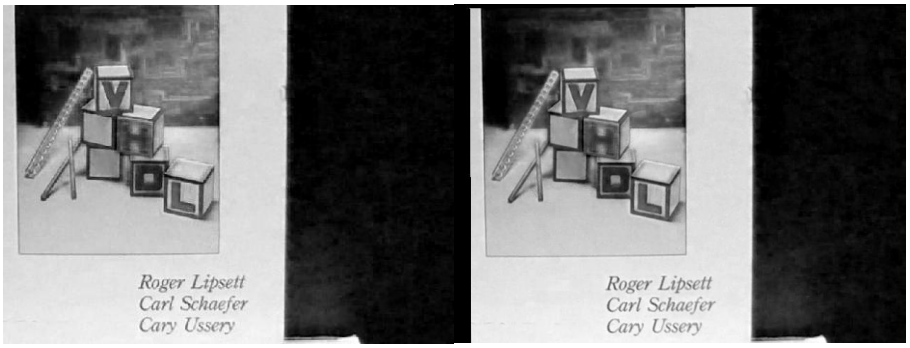
Figure 6-18: PSNR between stabilised frames and the reference image for both the original and the stabilised video sequence.



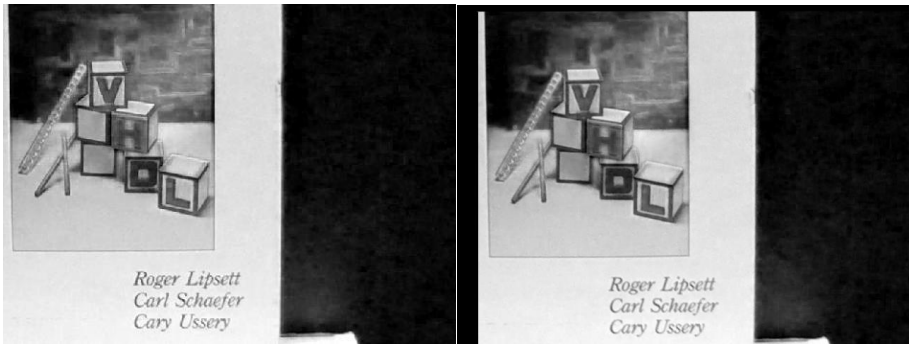
Frame20



Frame30



Frame40



Frame50

Figure 6-19: Left column: the original frames. Right column: the stabilised frames.

Figure 6-19 shows the stabilisation results, where the left and the right column shows the images before and after stabilisation, respectively. The processed video is stable, which indicates that the keypoints detected and matched using the system provide a solid basis for unwanted motion compensation.

6.4 Hardware Efficiency Evaluation

The design is fully embedded on a Xilinx XC6VLX240T FPGA device, which provides 301,440 registers, 150,720 LUTs, 768 DSP48E1 slices and 14.625Mbits BRAM blocks.

Table 6-1: FPGA resource usage for the whole system (VGA).

		Registers	**LUTs**	**BRAM (Mbits)**	**DSP48E1**	**Max Clock Frequency (MHz)**
Camera Controller		674	1,344	2.67	0	225
SIFT Processing core	**Feature Detection**	23,843	26,815	0.51	207	138
	Descriptor Generation	32,468	77,871	2.18	8	135
	Descriptor Matching[(2)]	3,568	15,662[(1)]	0	0	146
USB Controller		8,909	9,642	0	0	119
NPI Interface	**NPI Write**	3,243	3,436	0.59	0	202
	NPI Read	1,899	1,984	4.85	0	213
Whole System		92,748 (30.77%)	116,064 (77.01%)	11.74 (80.29%)	320 (41.47%)	115

(1) 4,096 of the total LUTs usage is configured as RAM holding coordinates of matched features.

6.4.1 Resource Usage

The design can be configured to process VGA and QVGA sized images, and the corresponding hardware resource usage of the whole system with the EDK platform included are given in Table 6-1 and

Table 6-2, respectively. The BRAM usage of *Camera Controller* includes buffer for input images. The BRAM usage of descriptor generation module includes the memory for generated descriptors. The descriptor matching module consists of two *Compare Descriptor* units, each of which consumes 1,244 registers and 4,368 LUTs.

Table 6-2: FPGA resource usage for the whole system (QVGA).

		Registers	**LUTs**	**BRAM (Mbits)**	**DSP48E1**	**Max Clock Frequency (MHz)**
SIFT Processing core	**Feature Detection**	23,756	26,387	0.33	207	138
	Descriptor Generation	32,392	77,863	1.11	8	135
	Descriptor Matching	3,477	12,787(1)	0	0	146
Whole System		89,128 (29.57%)	114,841 (76.19%)	5.17 (35.34%)	320 (41.47%)	115
(1) 2,048 of the total LUTs usage is configured as RAM holding coordinates of matched features.						

6.4.2 Comparison with the Existing Designs

This section compares the hardware efficiency of the design presented in this thesis with existing designs with respect to both processing speed and hardware resource usage.

Table 6-3: Hardware resource usage and throughput comparison of different hardware designs for feature detection (QVGA).

	[5]	**[55]**	**Proposed design**
Frame Size	QVGA		
Hardware Device/Technology	Altera Stratix II	Xilinx Virtex II Pro	Xilinx Virtex-6
Operating Frequency (MHz)	50	100	100
Registers	19,100	5,676	23,756
LUTs	43,366	5,554	26,387
DSP	64	N/A	207
Memory Usage (Mbits)	1.35	1.69	0.33
Frame Rate (fps)	30	900	306

Table 6-4: Hardware resource usage and throughput comparison of different hardware designs for feature detection (VGA).

	[50]	**[51]**		**[53]**	**Proposed design**
		High-Accuracy Mode	**High-Speed Mode**		
Frame Size	VGA				
Hardware Device/Technology	Xilinx Virtex-5	Altera Cyclone II		TSMC 0.18um	Xilinx Virtex-6
Clock Frequency (MHz)	100	50		100	100
Registers	19,529	23,247		N/A	23,843
LUTs	35,889	32,592			26,815
DSP	97	258			207
Memory Usage (Mbits)	3.24	0.87	0.67	0.896	0.51
Frame Rate (fps)	30	32	56	290	70

Table 6-3 and Table 6-4 show the comparison with some existing hardware designs for feature detection. When compared with [5] that processes QVGA images, the LUT usage and memory consumption of the system is reduced by approximately 39.2% and 75.6%, respectively. [55] implements only local extrema detection from DoG scale space, and does not include the keypoint refinement process and GMO computation. The keypoint refinement process involves complex matrix inversion, and GMO computation process contains arctan and square root computation, both of which are inefficient to be implemented on hardware devices. Therefore, [5] is not directly comparable with the proposed design. For implementation of local extrema detection from DoG scale space, the proposed design utilises 5,787 registers and 5,694 LUTs, which is virtually the same with that of [5]. When compared with [50], [51] and [53], which process VGA images, the design presented in this thesis has achieved memory reduction by approximately 84.3%, 23.9% (41.4% for high-speed mode), and 43.1%, respectively. The design has achieved the minimum memory usage as a result of the *rotating buffer* memory solution to Gaussian smoothed pixels and DoG values.

Table 6-5: Hardware resource usage and throughput comparison of different hardware designs for descriptor generation.

	[53]	**[57]**	**Proposed design**
Hardware Device/Technology	TSMC 0.13 um	TSMC 0.18 um	Xilinx Virtex-6
Clock Frequency (MHz)	200	100	100
Memory Usage (Mbits)	N/A	4.86	2.18
Time Consumption per Descriptor (us)	15.315	33.1	7.57
Descriptor Throughput	65,300	30,200	132,100

The performance comparison for descriptor generation, in terms of hardware resource usage and system throughput, is summarised in Table 6-5. With different hardware implementation technologies used, it is difficult to compare the resource

usage between different designs. However, the system throughput can be easily compared. It can be seen from Table 6-5 that the overall throughput of the proposed design is twice that of [53] and achieves speed improvement by approximately 4.37 times when compared with [57].

It should be noticed that the pipelined structure maximised the frame rate of the SIFT processing core, but the overall frame rate of the system is limited to half of the theoretical throughput of the SIFT processing core because of the data acquisition limit of the camera front-end. The camera can be configured to capture QVGA images at 60 fps or VGA images at 30 fps. However, this is not considered as a problem since the aim is to provide a high frame rate and high accuracy SIFT processing core.

6.5 Summary

In this chapter, an FPGA-based image matching system has been presented. The system has been designed and implemented in a Xilinx Virtex-6 FPGA device that includes the SIFT processing core, the interface to camera, the interface to USB, and the controller core for memory and data routing, which are all implemented using VHDL.

The SIFT processing core has achieved at least 60 VGA fps by using Xilinx ML605 FPGA board. However, the whole system with the camera front-end and the USB back-end included is not able to achieve this high throughput, which is limited by the camera front-end that captures grayscale images at 30 fps for VGA sized video and 60 fps for QVGA.

Tests of the SIFT-based image matching system have been conducted, from the robustness to geometric and photometric transformations, to the performance in applications such as object recognition and video stabilisation. The system can be configured to process QVGA or VGA images in two different modes to adapt to different applications. In the application for object recognition, the system works in the mode where input images are compared with the reference image of the target object or scene. In the application for video stabilisation, the system works in the mode where each input image is compared with the previous frame.

The whole system can capture images from the image sensor, run the SIFT-based processing step, and finally send data to a PC in real-time with high accuracy. In addition, as only 80% of the FPGA capacity is used, it is possible to add new image processing functions, if required by other applications.

Chapter 7 Summary, Conclusions and Discussion

7.1 Introduction

As stated in Chapter 1, the aim of this research project is to develop a high-performance real-time image matching system. Specifically, the following objectives have been addressed: (a) improvement towards the standard SIFT algorithm for an efficient hardware design; (b) high frame rate image matching system; (c) high accuracy matching system that achieves comparable performance with the software model; (d) low resource usage. The work carried out to fulfil these objectives has been presented in the previous chapters in this thesis. This chapter summarises the work that has been carried out throughput the project as a development step towards a high-performance real-time image matching system. Discussion to further optimise the system and suggestion for further work are also presented in this chapter.

7.2 Thesis Summary

Chapter 2 provided a basic introduction to related research into the intensity based feature detection methods that led to state-of-the-art SIFT algorithm. To improve either the efficiency or performance of the SIFT algorithm, many variations have been proposed, such as PCA-SIFT, SURF, GLOH and DAISY. DAISY has been proven to achieve the best result.

A review of systems aiming at accelerating SIFT was also carried out in Chapter 2, in terms of the processing aspect to improve the throughput of SIFT-based designs. The review showed that current researches mainly focus on the development of real-time feature detection part. However, little efforts have been made to improve the throughput of descriptor generation. Because SIFT has the potential of detecting a large number of features densely covering the entire image, descriptor generation process has become the bottleneck that would potentially prevent the entire system from achieving real-time, especially for systems that process high resolution images. This leads to the necessity of the research presented in this thesis.

Chapter 3 introduced SRI-DAISY, which is an alternative to the grid layout of the standard SIFT descriptor. The SRI-DAISY takes advantage of the log-polar spatial

arrangement of the standard DAISY, which is extended to be adaptive to image scaling and rotation. The performance of SRI-DAISY and the standard SIFT is compared for a wide range of transformations, including scaling, rotation, projection, blur and illumination changes. The SRI-DAISY achieves comparable performance with SIFT, but is more efficient as a result of the following aspects: (a) the memory requirement for buffering descriptors is reduced as a result of the dimension reduction from 128 to 72; (b) no need to shift all pixels within the local region; (c) no need to identify the boundaries of each sub-region.

A novel keypoint matching strategy was also presented in Chapter 3, which is inspired by the three existing widely applied matching methods. The novel matching strategy is superior to the distance ratio based matching in the following aspects: (a) achieve higher precision; (b) do not require hardware expensive square root computation or division operations.

In Chapter 4, design parameters that are essential to a high performance hardware design are studied. The design is parameterised with two octaves of five Gaussian smoothed images each. The system has been structured to compute the descriptors based on the pre-defined scales, which reduces both the memory requirement and processing time to a lower level at a cost of a little loss in matching performance. The fixed-point calculation is utilised to reduce the hardware resource usage. Experiments were conducted to determine the word length that is best balanced between computation accuracy and resource usage.

In Chapter 5, the FPGA-based processing core for the optimised SIFT is presented. All phases of the SIFT algorithm are covered: feature detection, descriptor generation and descriptor matching. The core utilises pipelined and parallel architecture to maximise the throughput. When running at 100 MHz in a Xilinx Virtex-6 FPGA, the processing core can achieve a frame rate of at least 60 fps for VGA images.

The feature detection utilises the SRT-based multi-pixel processing scheme and achieves at least 60 fps. The design can be modified to process images of higher resolution at a higher frame rate by making slight modification to the VHDL codes. Actually, in the current design, the overall throughput of feature detection is limited

by the speed at which pixels are accessed from the buffer holding input images. As discussed in Section 0, the input image buffer consists of two groups of RAM with two DPRAMs each, with which two pixels are accessed every clock cycle (5 ns). However, pixel throughput can be further increased by two means: (1) divide the input image into more parts with each loaded onto a separate DPRAM, thereby providing more ports to access pixels in parallel; (2) work with higher clock frequency. DPRAM supports a clock frequency of up to 450 MHz.

An efficient memory solution has been proposed in Chapter 5 for buffering Gaussian smoothed pixels and DoG values, named the *rotating buffer*. The *rotating buffer* is hardware efficient in the following aspect: the size is a constant and is independent of the number of pixels processed in parallel, which is beneficial when the design is modified to process more pixels for higher throughput.

Besides, an efficient hardware design for SRI-DAISY has been proposed in Chapter 5. The descriptor generation process takes advantage of the log-polar spatial arrangement and requires only 7.57 us to generate a descriptor of 72 dimensions, which is equivalent to a throughput of approximately 132,100 descriptors per second with a system clock of 100 MHz. When compared with existing hardware solutions, the design achieves the largest overall throughput with less hardware resource usage.

In Chapter 6, an embedded system was developed, which mainly consists of three parts: the camera front-end, the SIFT processing core presented in Chapter 5, and the USB back-end. Due to the data acquisition limit of the camera front-end, the processing core cannot run at its maximum available speed. The camera works at 30 fps for VGA, which limits the throughput of the entire system to 30 fps. Experimental results conducted on a set of real-world images were given to verify the functionality of the system. Besides, the system has been tested in two applications: object recognition and video stabilisation. The design is of high flexibility and can be configured to process QVGA or VGA images in two different modes to adapt to different applications.

7.3 Conclusions and Discussion

Throughout this thesis, the SIFT algorithm has been optimised and efficiently implemented to achieve the target of a high frame rate, high accuracy and low resource usage SIFT-based image matching system.

Although the design parameters have been selected to achieve the target of a high frame rate and high accuracy SIFT-based image matching system, some parameters, such as the amount of prior smoothing and descriptor matching threshold, can be modified to meet the requirement of different applications. When deciding the amount of prior smoothing and the size of the quantised Gaussian window, there is a trade-off between the distinctiveness and locality of the keypoints, which are the two competing properties that cannot be fulfilled simultaneously.

For applications such as image retrieval, where there are many candidate keypoints to be matched, detection regions identified by keypoints of lower locality contain more information and are easier to be correctly matched. However, these keypoints are more likely to suffer from geometric and photometric transformation. In the case of planar objects or in-plane rotation of camera, there is no occlusion or geometric transformation. The distinctiveness can be increased by increasing either the amount of prior smoothing or the size of the quantised Gaussian window. However, larger Gaussian window brings in higher computational complexity, more hardware area occupation and longer processing time for feature detection.

The quantity of the detected keypoints, which affects the performance of certain applications and the system throughput, is another property that needs to be taken into consideration when deciding the design parameters. Some applications require a large number of keypoints densely covering the objects of interest, such as object or scene recognition. However, a high number of keypoints has a negative impact on the computation time for descriptor generation, which is proportional to the number of descriptors to be generated and should be kept to a minimum. Decreasing the amount of prior smoothing or using quantised Gaussian window of smaller sizes contributes to an increased number of keypoints, which reduces the processing time of keypoints detection while increasing the processing time of descriptor generation.

Some applications are concerned more about the quality than the quantity of matching. For applications such as video stabilisation, the motion vectors are estimated based on the matched features by using model fitting methods, such as the least square or RANSAC. It has been stated in Chapter 3 that higher matching precision enables a model with higher accuracy and less processing time. Therefore, threshold values can be adjusted to improve the matching precision, such as decreasing the threshold for accepting matches with the ratio of $N_{\Delta d_{second}}$ to $N_{\Delta d_{closest}}$ below the pre-defined threshold. This will inevitably decrease the number of correct matches, but the matches are on the average more likely to be correct.

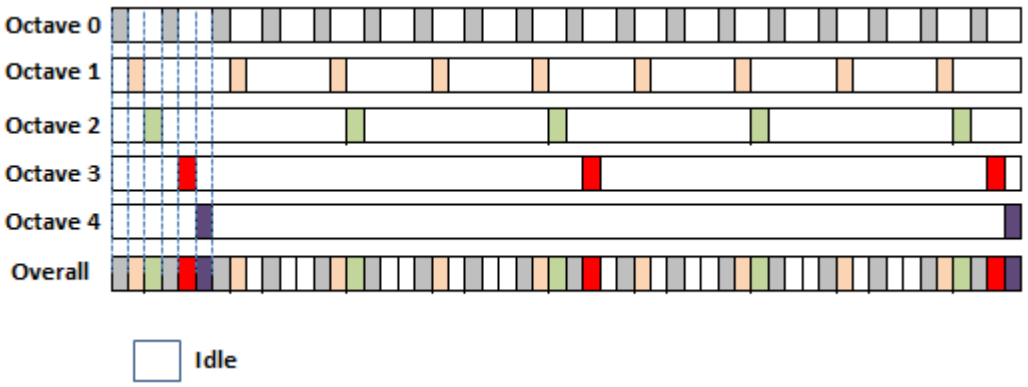


Figure 7-1: An example of octave interleaving with two clock cycles between adjacent Gaussian smooth process in the horizontal direction for octave 0.

In the current design, Gaussian pyramid construction is divided by octave and Gaussian blurred images within each octave are computed in parallel. When octave 0 has been processed, the Gaussian smoothed image is down-sampled spatially by a factor of two and acts as the input to the next octave. In the future, octave interleaving can be adapted, as illustrated in Figure 7-1. The idea is to make use of the clock cycles when the processing unit is in idle. For the example given in Figure 7-1, the gap between the Gaussian smoothing of adjacent pixels from octave 0 is two system clock cycles (10 ns), which supports five octaves to be interleaved without any two octaves requiring the same clock cycle. In the current design, because DPRAMs work with the clock domain of 200 MHz, whereas the SIFT processing core works with 100 MHz, it takes eight system clock cycles (10 ns) to access a

column of pixels for Gaussian smooth in the vertical direction using Gaussian kernel of size k_G=31. Therefore, the current design supports 17 octaves to be interleaved.

Octave interleaving is especially beneficial for processing images of higher resolution, such as UVGA images (1600x1200 pixels) that requires seven pixels to be processed in parallel to achieve real-time. However, the major disadvantage of octave interleaving is that extra RAMs are required to buffer intermediate results for different octaves, such as Gaussian smoothed pixels and DoG values, which is constant in the current design. Therefore, octave interleaving is suggested for designs with high availability in memory. But increasing the parallelism level by processing more pixels in parallel is recommended if memory availability becomes an issue. In this design, the hardware resource usage for processing one pixel is 5,787 registers and 5,694 LUTs.

In conclusion, throughout this thesis, a stand-alone image matching system was developed and tested successfully. This system can be widely used in computer vision related applications, such as Self Localisation and Mapping for robust navigation, 3D reconstruction, etc. The system also can be applied to applications beside computer vision, such as a real-time vision system for visual prosthesis simulator.

References

[1] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, “A comparison of affine region detectors,” *International Journal of Computer Vision*, vol. 65, no. 1/2, pp. 43–72, 2005.

[2] V. Ferrari, T. Tuytelaars, and L. Van Gool, “Simultaneous Object Recognition and Segmentation by Image Exploration,” *Proc. Eighth European Conf. Computer Vision*, pp. 40-54, 2004.

[3] D. G. Lowe, “Object recognition from local scale-invariant features,” I*nternational Conference on Computer Vision*, Corfu, Greece, pp. 1150-1157, 1991.

[4] S. Se, D. Lowe, and J. Little, “Global Localisation Using Distinctive Visual Features,” *Proc. Int’l Conf. Intelligent Robots and Systems*, pp. 226-231, 2002.

[5] V. Bonato, E. Marques, and G. A. Constantinides, “A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection,” *IEEE Transactions on Circuits and Systems for VideoTechnology*, Vol. 18, NO. 12, Dec. 2008.

[6] H. Grabner, J. Matas, L. Van Gool, and P. Cattin, “Tracking the invisible: Learning where the object might be,” *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp.1285-1292, June 2010.

[7] S. Tran, L. Davis, “Robust Object Tracking with Regional Affine Invariant Features,” *Computer Vision, 2007.ICCV 2007. IEEE 11th International Conference on* , pp.1-8, Oct. 2007.

[8] N. Snavely, R. Garg, S. M. Seitz, and R. Szeliski, “Finding Paths through the World's Photos,” In *ACM Transactions on Graphics (TOG), SIGGRAPH '08*, 2008.

[9] R. Hess, A. Fern, "Improved Video Registration using Non-Distinctive Local Image Features," *Computer Vision and Pattern Recognition, 2007.CVPR'07. IEEE Conference on*, pp.1-8, June 2007.

[10] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *Int. J. Computer Vision.*, vol. 60, no. 2, pp. 91-110, Jan. 2004.

[11] S. Lazebnik, C. Schmid, and J. Ponce, “Sparse texture representation using affine invariant neighborhoods,” in *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pp. 319–324, 2003.

[12] K. Mikolajczyk, B. Leibe, and B. Schiele, “Multiple object class detection with a generative model,” in *Proceedings the Conference on Computer Vision and Pattern Recognition*, pp. 26–36, 2006.
[13] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *Proceedings of the International Conference on Computer Vision*, pp. 1470–1478, 2003.
[14] H. Freeman and L. S. Davis, “A corner-finding algorithm for chain-coded curves,” *IEEE Transactions on Computers*, vol. 26, pp. 297–303, 1977.
[15] H. Moravec, “Rover visual obstacle avoidance,” in *International Joint Conference on Artificial Intelligence*, Vancouver, Canada, pp. 785-790, 1981.
[16] C. Harris and M. Stephens, “A combined corner and edge detector,” in *Alvey Vision Conference*, pp. 147–151, 1988.
[17] K. Mikolajczyk and C. Schmid, “Scale and affine invariant interest point detectors,” *International Journal of Computer Vision*, vol. 1, no. 60, pp. 63–86, 2004.
[18] T. Lindeberg, “Feature detection with automatic scale selection,” *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, 1998.
[19] P. R. Beaudet, “Rotationally invariant image operators,” in *Proceedings of the International Joint Conference on Pattern Recognition*, pp. 579–583, 1978.
[20] T. Lindeberg, “Feature detection with automatic scale selection,” *International Journal of Computer Vision*, 30(2):79–116, 1998.
[21] A. P. Witkin, “Scale-space filtering,” *In International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 1019-1022, 1983.
[22] J. J. Koenderink, “The structure of images,” *Biological Cybernetics*, 50:363-396, 1984.
[23] T. Lindeberg, “Scale-space theory: A basic tool for analysing structures at different scales,” *Journal of Applied Statistics*, 21(2):224-270, 1994.
[24] J. L. Crowley and A. C. Parker, “A representation for shape based on peaks and ridges in the difference of low pass transform,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 2, pp. 156–170, 1984.
[25] P. Gaussier and J. P. Cocquerez, “Neural networks for complex scene recognition: Simulation of a visual system with several cortical areas,” in

Proceedings of the International Joint Conference on Neural Networks, vol. 3, pp. 233–259, 1992.

[26] S. Grossberg, E. Mingolla, and D. Todorovic, "A neural network architecture for preattentive vision," *IEEE Transactions on Biomedical Engineering,* vol. 36, pp. 65–84, 1989.

[27] K. Mikolajczyk, and C. Schmid, "A performance evaluation of local descriptors," *PAMI 27*, pp. 1615–1630, 2005.

[28] Y. Ke and R. Sukthankar. "PCA-SIFT: A More Distinctive Representation for Local Image Descriptors," *Proc. Conf. Computer Vision and Pattern Recognition*, pp. 511-517, 2004.

[29] I. T. Joliffe. "Principal Component Analysis," Springer-Verlag, 1986.

[30] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the Conference on Computer Vision andPattern Recognition*, vol. 1, pp. 511–518, 2001.

[31] C. Crow. Franklin, "Summed-area Tables for Texture Mapping," in *Proceedings of the 11th annual conference on Computer graphiecs and interactive techniques*, pp. 207-212, 1984.

[32] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision*, pp. 404–417,2006.

[33] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features(SURF)," *International Journal on Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[34] C. K. Chui, L. Montefusco, and L. Puccio, "Wavelet: Theory, algorithms, and applications," *Academic Press,* San Diego, CA, 1994.

[35] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," in *Proc. CVPR*, 2008.

[36] S.A.J. Winder, and M. Brown, "Learning Local Image Descriptors," *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp.1-8, June 2007.

[37] Q. Zhang, Y. Chen, Y. Zhang, and Y. Xu, "SIFT implementation and optimization for multi-core systems," *Parallel and Distributed Processing, 2008. IPDPS'08. IEEE International Symposium on*, pp.1-8, 2008.

[38] H. Feng, E. Li, Y. Chen and Y. Zhang, "Parallelization and characterization of SIFT on multi-core systems," *Workload Characterization, 2008. IISWC'08. IEEE International Symposium on*, pp.14-23, Sept. 2008.

[39] S. N. Sinha, J. M. Frahm, M. Pollefeys, and Y.Genc, "Feature Tracking and Matching in Video Using Programmable Graphics Hardware", *Machine Vision and Application*, 2007.

[40] S. Heymann, B. Frohlich, F. Medien, K. Muller, and T. Wiegand, "SIFT Implementation and Optimization for General-propose GPU," in *Proc. WSCG'07* ,pp.317-322, 2007.

[41] C. Wu, "SiftGPU: A GPU implementation of scale invariant feature transform (SIFT)," Available at http://cs.unc.edu/~ccwu/siftgpu/.

[42] B. Cope, P.Y.K. Cheung, W. Luk, and S. Witt, "Have GPUs made FPGAs redundant in the field of video processing?," in *Proc. of the IEEE Int. Conf. on Field-Prog. Technology*, pp. 111-118, 2005.

[43] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance comparison of FPGA, GPU and CPU in image processing," in P*roc. of Int. Conf. on Field Prog, Logic and App. FPL '09*, pp.126-131, 2009.

[44] J. Bodily, B. Nelson, Z. Wei, D. Lee, and J. Chase, "A comparison study on implementing optical flow and digital communications on FPGAs and GPUs," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, pp. 1-22, 2010.

[45] J. Fowers, G. Brown, J. R. Wernsing, and G. Stitt, "A performance and energy comparison of convolution on GPUs, FPGAs, and multicore processors," *TACO*, 2013.

[46] S. Se, H.-K.Ng, P. Jasiobedzki, and T.-J.Moyung, "Vision based modelling and localisation for planetary exploration rovers," in *Proceedings of International Astronautical Congress (IAC)*, Vancouver (Canada), 2004.

[47] N. Pettersson, and L. Petersson, "Online stereo calibration using FPGAs," *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pp. 55-60, June 2005.

[48] H. D. Chati, F. Muhlbauer, T. Braun, C. Bobda, and K. Berns, "Hardware/Software co-design of a keypoint detector on FPGA," *Field-*

Programmable Custom Computing Machines, 2007.FCCM 2007. 15th Annual IEEE Symposium on , pp.355-356, Apr. 2007.

[49] S. Thrun, W. Burgard, and D. Fox, "*Probabilistic Robotics*," *Cambridge, USA: MIT Press,* 2005.

[50] L. Yao, H. Feng, Y. Zhu, Z. Jiang, D. Zhao, and W. Feng, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," *Field-Programmable Technology, 2009. FPT'09. International Conference on* , pp.30-37, Dec. 2009.

[51] K. Mizuno, H. Noguchi, G. He, Y. Terachi, T. Kamino, H. Kawaguchi, and M. Yoshimoto, "Fast and Low-Memory-Bandwidth Architecture of SIFT Descriptor Generation with Scalability on Speed and Accuracy for VGA Video," *Field Programmable Logic and Applications (FPL), 2010 International Conference on* , pp.608-611, 2010.

[52] E. S. Kim, and H. -J. Lee, "A practical hardware design for the keypoint detection in the SIFT algorithm with a reduced memory requirement," *Circuits and Systems (ISCAS), 2012 IEEE International Symposium on* , pp.770-773, May 2012.

[53] F. Huang, S. Huang, J. Ker, and Y. Chen, "High-Performance SIFT Hardware Accelerator for Real-Time Image Feature Extraction," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.22, no.3, pp.340-351, Mar. 2012.

[54] B. Bosi, G. Bois, and Y. Savaria, "Reconfigurable pipelined 2-D convolvers for fast digital signal processing," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 7, no. 3, pp. 299–308, Mar. 1999.

[55] L. Chang, J. Hernandez-Palancar, L. Enrique Sucar and M. Arias-Estrada, "FPGA-based detection of SIFT interest keypoints," Machine Vision and Applications, vol. 24, pp. 371-392, 2013.

[56] L. Chang, and J. Hernandez-Palancar, "A Hardware Architecture for SIFT Candidate Keypoints Detection," CIARP, Berlin, Germany, pp. 95-102, 2009.

[57] Y. Lin, C.Yeh, S. Yen, C. Ma, P. Chen, and C.-C.J. Kuo, "Efficient VLSI design for SIFT feature description," *Next-Generation Electronics (ISNE), 2010 International Symposium on* , pp.48-51, Nov. 2010.

[58] S.A.J. Winder, G. Hua, and M. Brown. “Picking the best DAISY,” In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 0, pages 178–185, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[59] E. Tola, V. Lepetit, and P. Fua, “A fast local descriptor for dense matching,” in Proc. CVPR, Citeseer, 2008.

[60] Fischer, J.; Ruppel, A.; Weisshardt, F.; Verl, A., “A rotation invariant feature descriptor O-DAISY and its FPGA implementation,” *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on* , vol., no., pp.2365,2370, 25-30 Sept. 2011.

[61] Feature detector evaluation sequences. Available at http://lear.inrialpes.fr/people/Mikolajczyk.

[62] M. Stephenson, J. Babb, and S. Amarasinghe, “Bitwidth analysis with application to silicon compilation,” in *Proceedings of the SIGPLAN conference on Programming Language Design and Implementation*, June 2000.

[63] M. L. Chang, S. Hauck, “ Least-significant bit optimization techniques for FPGAs,” in *Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, FPGA 2004*, USA, pp. 251, 2004.

[64] Xilinx, “Virtex-6 FPGA DSP48E1 Slice User Guide”, Feb 2011.

[65]General architecture of an FPGA device. Available at www.xilinx.com.

[66] Xilinx, “Dual-Port Block Memory Core (v6.3),” 2005.

[67] Gustavo Carneiro, Allan D. Jepson, “The quantitative characterization of the distinctiveness and robustness of local image descriptors,” Image and Vision Computing, Volume 27, Issue 8, 2 July 2009, Pages 1143-1156, ISSN 0262-8856.

[68] K. Mikolajczyk, C. Schmid, “Indexing based on scale invariant interest points,” in: *IEEE International Conference on Computer Vision*, Vancouver, Canada, pp. 525–531, 2001.

[69] Xilinx, “LogiCORE IP Multi-Port Memory Controller (MPMC) (v6.03.a),” 2011.

[70] Xilinx, “ML605 Hardware User Guide (v1.8),” Oct. 2012.

[71] Xilinx, “EDK Concepts, Tools, and Techniques,” Jan. 2012.

[72] Morimoto, C.; Chellappa, R., "Evaluation of image stabilization algorithms," *Acoustics, Speech and Signal Processing, 1998. Proceedings of the*

1998 IEEE International Conference on , vol.5, no., pp.2789,2792 vol.5, 12-15 May 1998.

Appendix A. Extrema Detection with Stability Checking

Gaussian scale space consists of two octaves with five scales each. By comparing pixels with their neighbours, local extrema belonging up to two scales are detected. A pixel will be passed to the stability checking process if it is a local extremum and will be identified as a keypoint after is has passed through the three refinement processes. Although two pixels are processed in parallel for higher system throughput, only one stability checking module is used because seldom has the chance that two pixels lie next to each other are both extrema. Each local extremum detected from the DoG scale space is passed to the stability checking process, which consists of three steps: location refinement, low contrast removal, and edge response elimination. The overall hardware structure is shown in Figure A-1, where three sub-modules have no data dependency and are processed in parallel by taking advantage of the parallel processing property of FPGA.

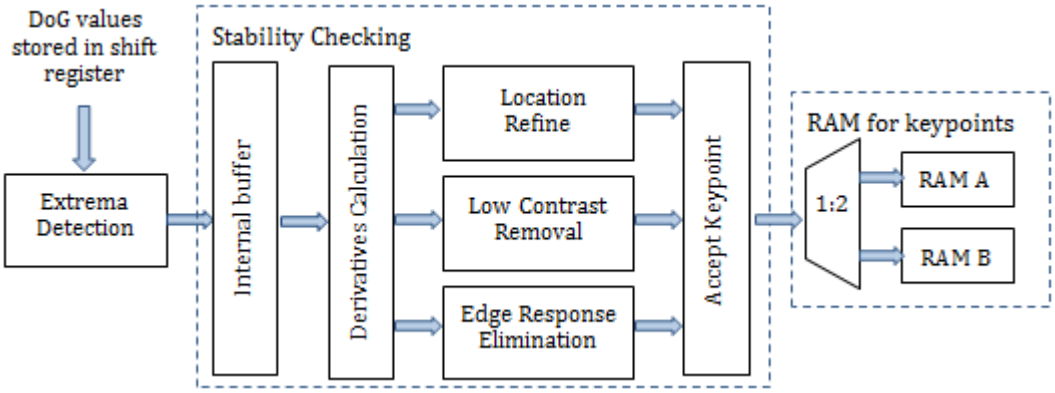


Figure A-1: Block diagram for extrema detection with stability checking.

The minimum throughput requirement (TPR_{ED}) of the extrema detection block is shown in Equation (A.1). Because only those pixels that are local extrema are passed to stability checking block, TPR_{ED}actually corresponds to the maximum throughput requirement of stability checking process.

$$TPR_{ED} = 2f \cdot \sum_{i=0}^{n_{oct}} [(w_i - k_G - 1)(h_i - k_G - 1)] \quad \text{(A.1)}$$

where f is the system frame rate (60 fps).

As a result of the SRT-based multi-pixel streaming scheme, two Gaussian smoothed pixels are generated every $\frac{k_G+1}{4}$ clock cycles of 100 MHz after an initial delay. To keep a constant overall throughput for feature detection module, it is suggested that the extrema detection and stability checking process should finish within $\frac{k_G+1}{4}$ clock cycles. Because the extrema detection sub-module includes only 26 simple comparison operations, it is easy to be completed within a few clock cycles with hardware parallel property explored. So the design mainly focuses on the solution to the stability checking block and uses internal buffer and registers to create pipelined architecture. As shown in Figure A-1, an internal buffer is inserted between the extrema detection and the stability checking process for two purposes:

1. The stability checking process is carried out based on the DoG values stored in the internal buffer and has no direct data dependency with the extrema detection process.
2. In the most unlikely cases that both pixels processed in parallel are local extrema, the related neighbouring DoG values can be stored in the internal buffer before the previous pixel has been processed.

With the internal buffer, the extrema detection with stability checking module is arranged into a two-stage pipelined architecture. With intermediate computation results within the stability checking process registered by the clock, the pipelined architecture is adopted within the stability checking process, making it possible to deal with consecutively arrived extrema and the time requirement of the stability checking process is no longer of a great concern.

Appendix B. NPI PIM Interface

NPI PIM Write Interface

The interface of NPI PIM Write Unit is shown in Figure B-1. The NPI PIM write unit has been developed to support 64-bit NPI 32-word burst write, and the description of the interface is given in Table B-1. MPMC_Clk0 is the main MPMC clock and is set to 200 MHz.

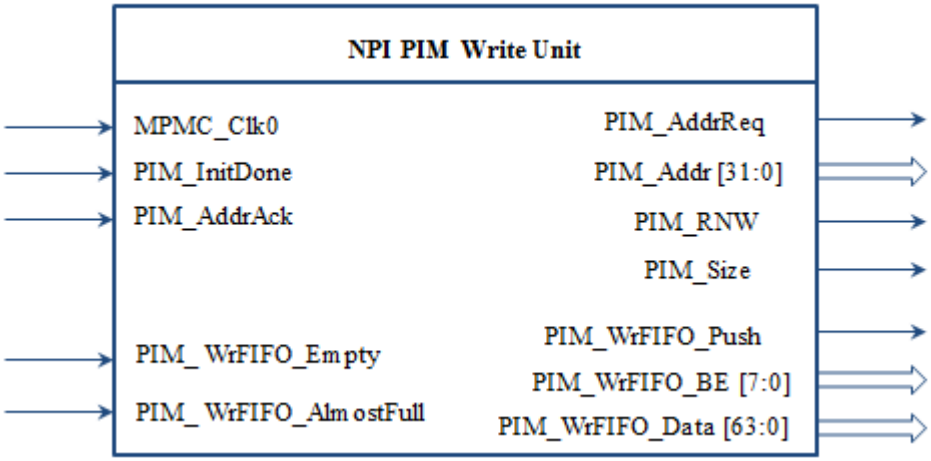


Figure B-1: NPI PIM write interface with 64-bit NPI 32-word burst write.

Table B-1: Signals related to NPI PIM write interface and their functions.

Signal Name	Description
MPMC_Clk0	Clock signal.
PIM_InitDone	'1' indicates that initialisation is complete and that FIFOs are available for use. Do not assert PIM_WrFIFO_Push until PIM_InitDone is equal to '1'.
PIM_AddrAck	This active high signal indicates that MPMC has begun arbitration for address request. Valid for one cycle of MPMC_Clk0.
PIM_WrFIFO_Empty	This active high signal indicates that there are less than 32 bits of data in the write FIFO.
PIM_WrFIFO_AlmostFull	This active high signal indicates that PIM_WrFIFO_Push cannot be asserted on the next cycle of MPMC_Clk0. This signal is only asserted when using SRL FIFOs. If BRAM FIFOs are used, the PIM cannot allow more than 1024 bytes of data to be pushed into the FIFOs.
PIM_AddrReq	This active high signal indicates that NPI is ready for MPMC to arbitrate an address request. This request cannot be aborted. Must be asserted until PIM_AddrAck is asserted.
PIM_Addr	Indicates the starting address of a particular request. Only valid when PIM_AddrReq is valid. Must be aligned to Size burst length.
PIM_RNW	Read/Not Write: 0 = Request is a Write request. 1 = Request is a Read request. Only valid when PIM_AddrReq is valid.
PIM_Size	Indicates the transfer type of the request: 0x4 = 32-word burst transfers Only valid when PIM_AddrReq is valid.
PIM_WrFIFO_Push	This active high signal indicates push WrFIFO_Data into write FIFOs. Cannot be asserted while PIM_InitDone is 0. Cannot be asserted while WrFIFO_AlmostFull is asserted. Can be asserted before, after, or during the address phase unless MPMC is configured in one of several special cases.
PIM_WrFIFO_BE	Indicates which bytes of WrFIFO_Data to write. Only valid with PIM_WrFIFO_Push.
PIM_WrFIFO_Data	Data to be pushed into MPMC write FIFOs. Only valid with PIM_WrFIFO_Push.

NPI PIM Read Interface

The entity of NPI PIM Read Unit is shown in Figure B-2.

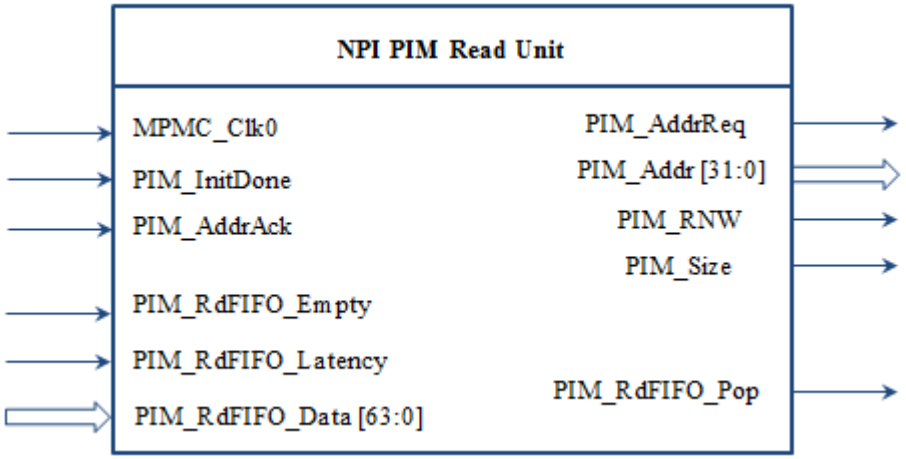


Figure B-2: NPI PIM read interface with 64-bit NPI 32-word burst read.

The NPI PIM read unit has been developed to support 64-bit NPI 32-word burst read, and the description of the interface is displayed in Table B-2. Signals shared between NPI PIM write interface and read interface are not repeated here.

Table B-2: Signals related to NPI PIM read interface and their functions.

Signal Name	Description
PIM_RdFIFO_Empty	When this active high signal is de-asserted, it indicates that enough data is in the read FIFOs to assert PIM_RdFIFO_Pop.
PIM_RdFIFO_Latency	Indicates the number of cycles from the time PIM_RdFIFO_Pop is asserted and/or PIM_RdFIFO_Empty is de-asserted until PIM_RdFIFO_Data is valid
PIM_RdFIFO_Pop	This active high signal indicates that read FIFO fetch the next value of PIM_RdFIFO_Data. Must be asserted for one cycle of MPMC_Clk0.
PIM_RdFIFO_Data	Data to be popped out of MPMC read FIFOs.

Appendix C. Hardware Architecture of the Descriptor Generation Module

Gaussian Weighting Factor Controller

As shown in Figure 5-26, each sub-region is defined as a rectangle of size $D \times D$ (D is the diameter of the circular sub-region) for simplicity. To generate the gradient histogram for each sub-region, gradient magnitude of each pixel within the sub-region has to be weighted by Gaussian function with the parameter of the distance from the pixel to be weighted and the centre of the corresponding sub-region. Pixels located closer to the centre offer larger contribution to the sub-region histogram, and pixels outside the circular sub-region offer no contribution by setting the corresponding weighting factors to 0. With the Gaussian function used to weight the gradient magnitude of pixels within each sub-region, the square sub-regions can be regarded as circular ones. Considering the isotropy character of both the circular sub-region and Gaussian function, contribution of a pixel will be the same however the image rotates, because the distance is not changed.

Because the Gaussian weighting factors concern only the distance from pixels to be weighted to the centre of the corresponding circular region, they can be calculated offline and pre-loaded into an LUT. The LUT is an array of values used to reduce processing time for applications that uses complex calculations, which is an efficient alternative to the complex computations. An LUT holds data or results calculated offline from the complex calculations needed by applications, and gives an output value for each index value. By keeping the results in the LUT, data can be accessed immediately by referring to the LUT instead of doing calculations, and the complex computation is replaced by simpler array indexing operations. Therefore, the LUT is an optimal choice for reducing the computational complexity and processing time of hardware designs.

The block diagram of the Gaussian Weighting Factor Controller is shown in Figure C-1. In Figure C-1, (x_c, y_c) is the centre coordinates of the sub-region being processed, and (x_i, y_i) represents the coordinates of pixels within the sub-region.

Figure C-1 shows that a Gaussian weighting factor can be identified with two subtract operations and an LUT within two clock cycles, which greatly reduce the computational complexity and the processing time. Taking advantage of the symmetrical property of Gaussian filter, only a quarter of the entire Gaussian window is loaded into the LUT to further save the memory.

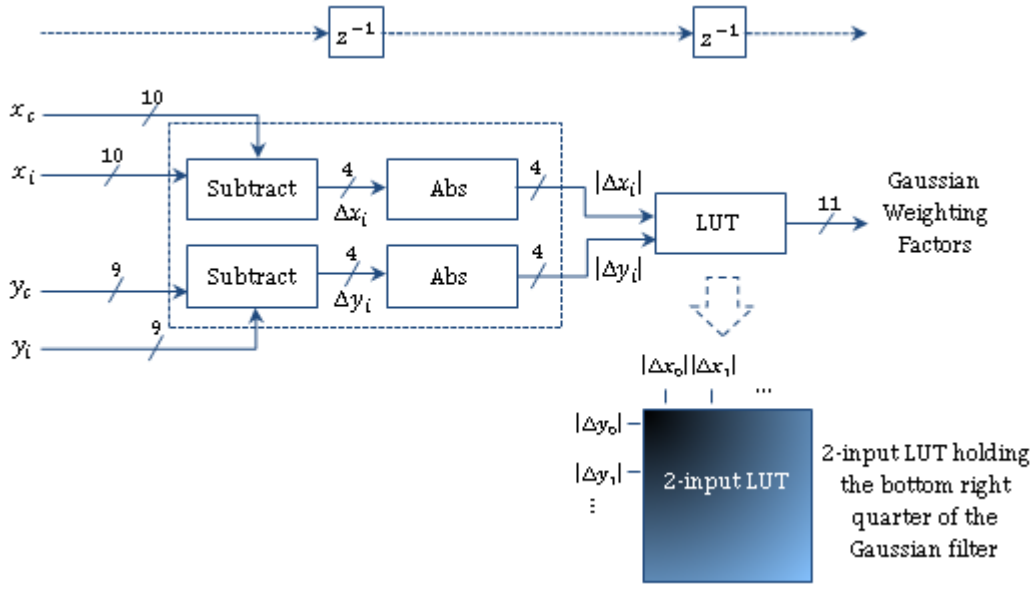


Figure C-1: Gaussian weighting factor controller with 2-input LUT.

Principal Orientation Calculation

In this sub-module, pixel values within each sub-region are weighted and accumulated to generate the 36-bin gradient histogram. The first step of the descriptor generation is to identify the principal orientation (θ_{po}), which corresponds to the orientation of the largest bin in the histogram obtained by weighting and accumulating all pixels within the local region. Figure C-2 shows the block diagram of the Principal Orientation Calculation module.

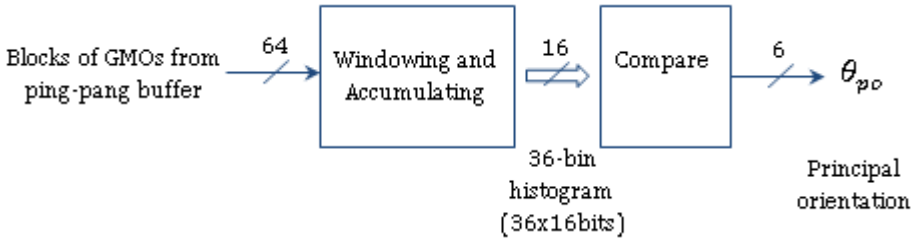


Figure C-2: Block diagram for the Principal Orientation Calculation module (pipeline stage 1).

To make full use of the throughput of the DPRAM, two blocks of GMOs are accessed per clock cycle. A block is actually four sets of GMOs that are concatenated and buffered in DDR3 as a single data. Taking advantage of the parallel processing property of FPGA, these two blocks are processed in parallel to increase the throughput. Figure C-3 shows the block diagram of the Windowing and Accumulating unit. The GMO blocks routed from ping pong buffers are split into four sets of GMOs and then sent to four Processing Units (PUs).

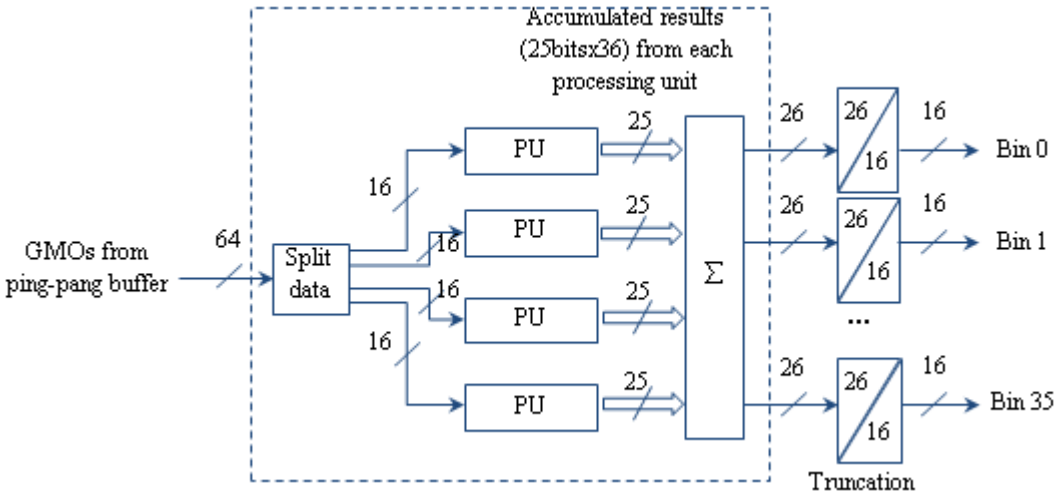


Figure C-3: Block diagram for the Windowing and Accumulating Unit.

The block diagram of the PU is shown in Figure C-4.

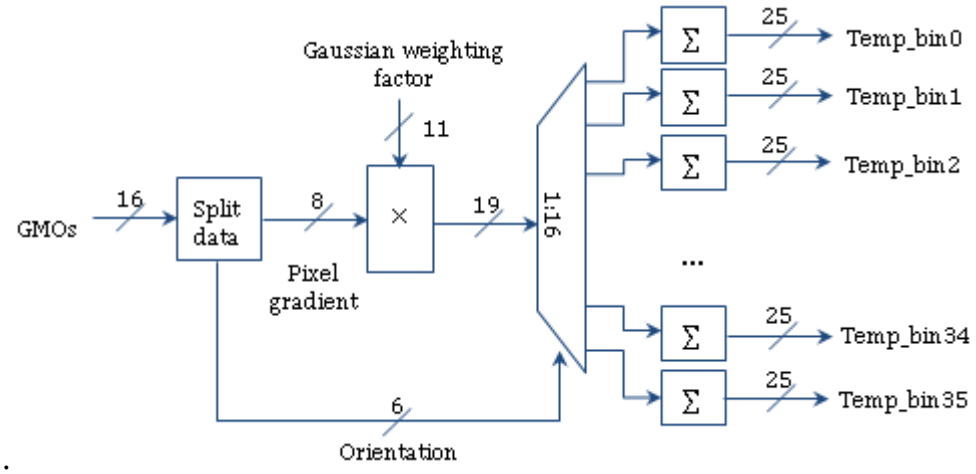
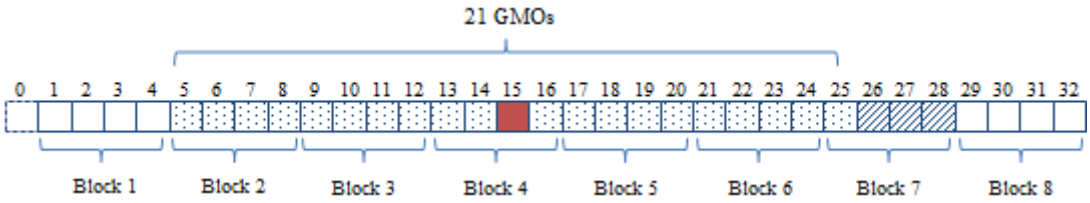


Figure C-4: Detailed architecture of the PU.

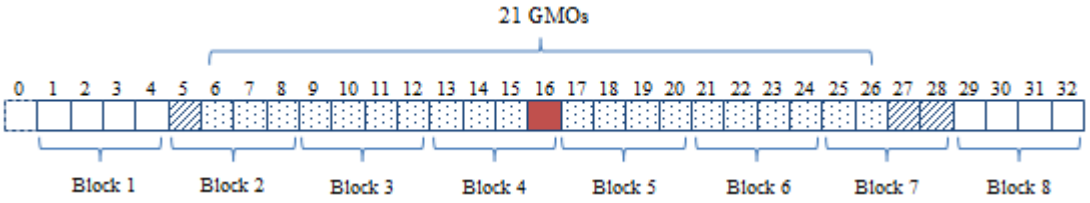
For keypoints from scale2 and scale3, the diameter of circular sub-regions is 15 and 21, respectively. With four sets of GMOs grouped as a single data block, 4x15 blocks of GMOs are required for each sub-region from scale2 and 6x21 blocks for scale3. Only 21 GMOs per row are needed for keypoints from scale 3, but 6 blocks give 24 sets. Therefore, three sets of GMOs have to be abandoned. The idea is to divide the x coordinate of the first set of valid data by four, retaining the remainder as the mode selector that decides which sets to be discarded, as shown in Table C-1.

Table C-1: Relationship between remainder and mode for keypoints from scale3.

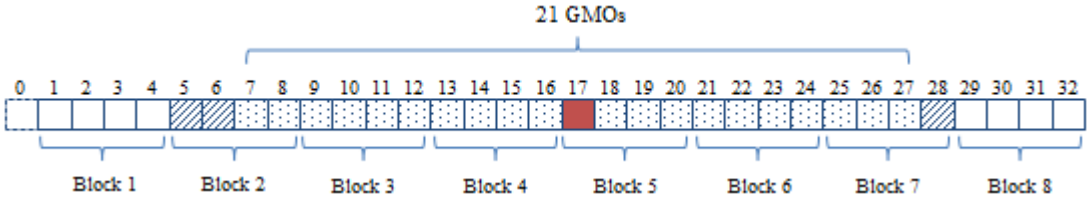
Remainder	**Mode**
1	0-3
2	1-2
3	2-1
0	3-0



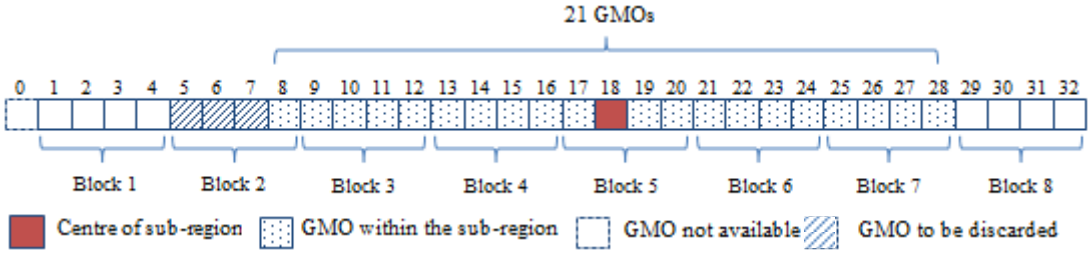
(a) Mode 0-3.



(b) Mode 1-2.



(c) Mode 2-1.



(b) Mode 3-0.

Figure C-5: Four different circumstances to discard certain data.

Figure C-5 shows four different modes in which certain sets need to be discarded. Considering that divide-by-four is equivalent to right-shifting the signal by two bits, the last two bits of the x coordinate of the first set of GMOs is equivalent to the remainder and is used as the mode selector. Figure C-6 describes this idea by using Mode 1-2 shown in Figure C-5(b) as an example. As a result, the division operation can be avoided in this processing unit.

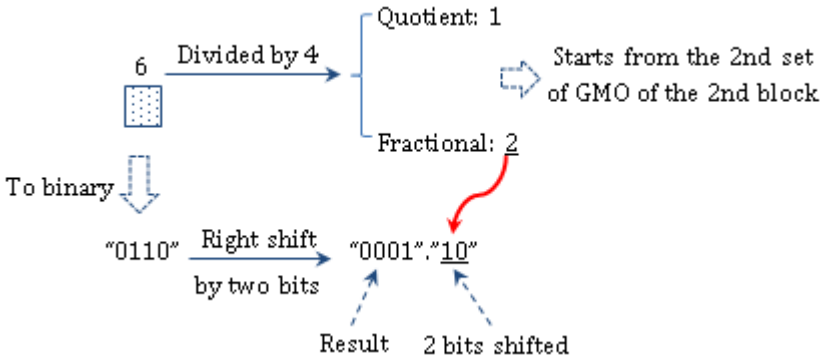


Figure C-6: The mode selector for GMO access.

For keypoints from scale2, four blocks gives 16 GMOs. Either the first or the last set of GMOs has to be discarded. In this case, the remainder from right shifting the x coordinate by two acts as the mode selector, as shown in Table C-2.

Table C-2: Relationship between remainder and mode for keypoints from scale2.

Remainder	**Mode**
0	1-0
1	0-1

Centre Coordinates Calculation

Figure C-7 shows the block diagram of the Centre Coordinates Calculation unit, which consists of an LUT and two signed adders. This unit inputs both the principal orientation θ_{po} and the coordinates (x_c,y_c) of the keypoint, and outputs the centre coordinates of eight surrounding sub-regions (x_{ORi},y_{ORi}), where i is the index to the eight surrounding sub-regions and is in range 0 to 7. Again, the LUT technique is employed to avoid the complex sin and cos operations.

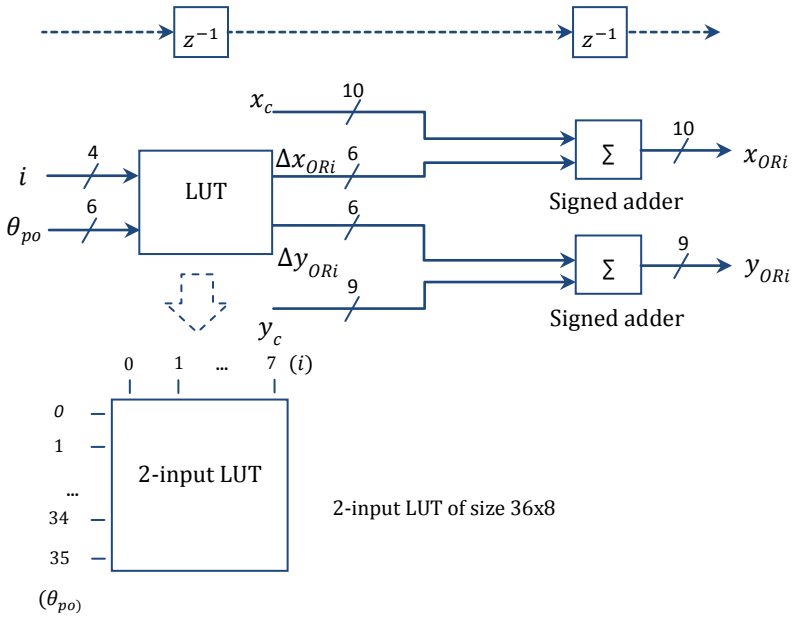


Figure C-7: Block diagram of the Centre Coordinates Calculation unit with a 2-input LUT (pipeline stage 2).

Table C-3: Centre coordinates of circular sub-regions relative to the feature point in both x and y directions.

Sub-region	Δx_{ORi}	Δy_{ORi}
1	$R\cos(\theta_{po})$	$R\sin(\theta_{po})$
2	$\frac{\sqrt{2}}{2}R\left(\cos(\theta_{po})-\sin(\theta_{po})\right)$	$\frac{\sqrt{2}}{2}R\left(\cos(\theta_{po})+\sin(\theta_{po})\right)$
3	$-R\sin(\theta_{po})$	$R\cos(\theta_{po})$
4	$-\frac{\sqrt{2}}{2}R\left(\cos(\theta_{po})+\sin(\theta_{po})\right)$	$\frac{\sqrt{2}}{2}R\left(\cos(\theta_{po})-\sin(\theta_{po})\right)$
5	$-R\cos(\theta_{po})$	$-R\sin(\theta_{po})$
6	$\frac{\sqrt{2}}{2}R\left(\sin(\theta_{po})-\cos(\theta_{po})\right)$	$-\frac{\sqrt{2}}{2}R\left(\cos(\theta_{po})+\sin(\theta_{po})\right)$
7	$R\sin(\theta_{po})$	$-R\cos(\theta_{po})$
8	$\frac{\sqrt{2}}{2}R\left(\cos(\theta_{po})+\sin(\theta_{po})\right)$	$\frac{\sqrt{2}}{2}R\left(\sin(\theta_{po})-\cos(\theta_{po})\right)$
9	0	0

In Table C-3, (Δx_{ORi},Δy_{ORi}) represent the offsets from the centre coordinates of eight surrounding sub-regions (x_{ORi},y_{ORi}) to that of the keypoint (x_c,y_c) in both x and y directions. It can be seen from Table C-3 that coordinates offsets (Δx_{ORi},Δy_{ORi}) are only related to θ_{po} and R, where R is the distance between the centre pixel of surrounding sub-regions and the keypoint. Considering that R is fixed and θ_{po} has been normalised to integers in range 0 to 35, (Δx_{ORi},Δy_{ORi}) can be calculated offline and pre-loaded onto a single-input LUT with θ_{po} acting as the index.

It has been mentioned that redundantly rotating the coordinates of all pixels within the local region for rotation invariance has been replaced by arranging both the location and the 2D gradient histogram of each sub-region relative to the principal orientation. With the LUT-based coordinate calculation method, the location of each surrounding sub-regions is arranged relative to the principal orientation by using only two adders, as shown in Figure C-7. As a result, the rotation invariance of sub-region arrangement can be achieved with adders, and the hardware expensive sin and cos operations are avoided.

36-bin Histogram Calculation

Figure C-8 shows the block diagram of the 36-bin Histogram Calculation, where pixel values within each sub-region are weighted and accumulated to generate the 36-bin gradient histograms. The Windowing and Accumulating unit shares the same architecture with that shown in Figure C-3, but with different PU. As shown in Figure C-9, pixel orientation is normalised relative to the principal orientation to achieve rotation invariance.

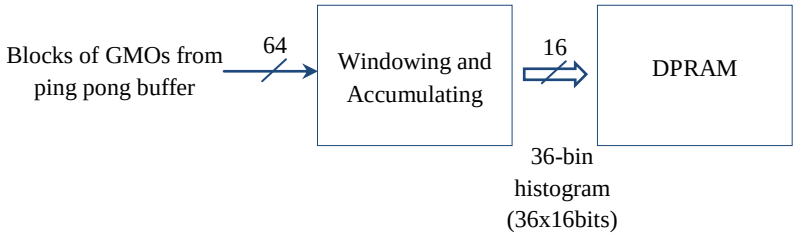


Figure C-8: Block diagram for 36-bin Histogram Calculation (pipeline stage 3).

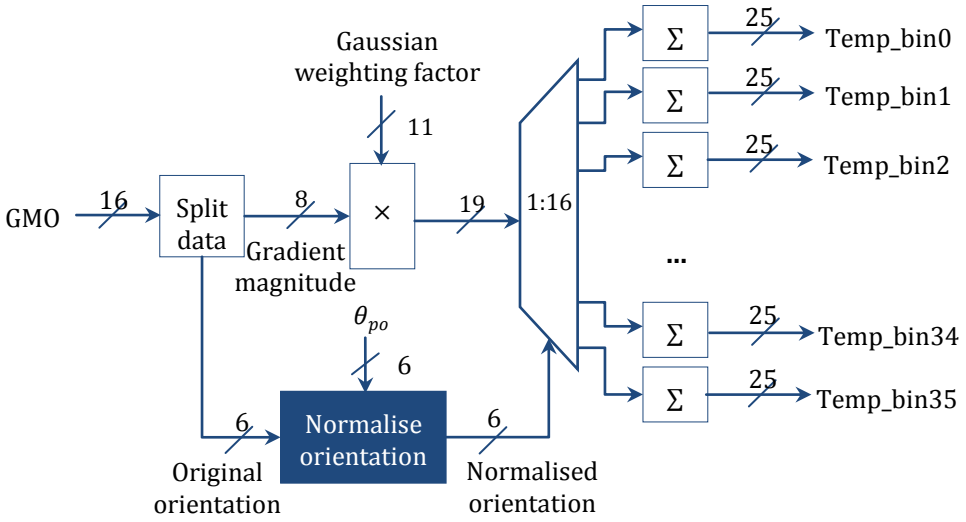


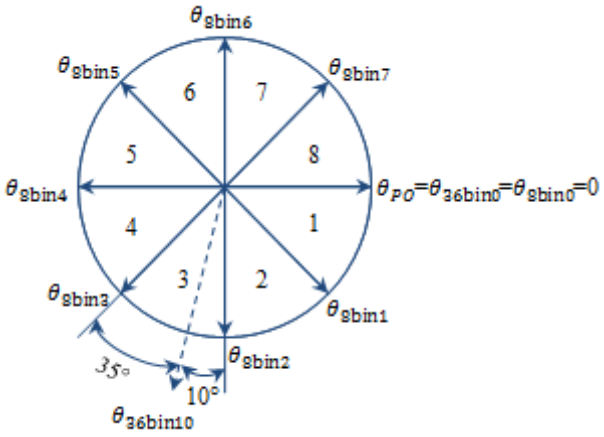
Figure C-9: Block diagram for PU of 36-bin histogram calculation.

Linear Interpolation

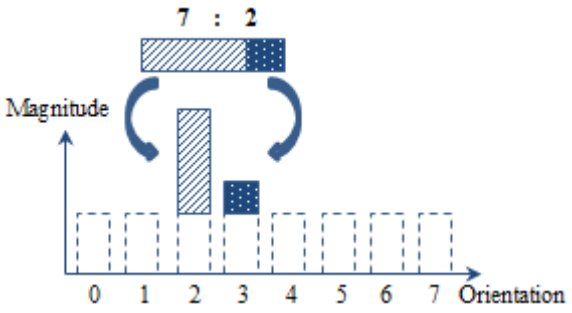
This sub-module inputs the 36-bin histogram and outputs the interpolated 8-bin histogram with each bin representing 45 °. The linear interpolation is realised base on

$$Bin36_{i\_j} = Bin36_i \times F_{i\_j} \tag{C.1}$$

where i is the index to 36 bins of the input 36-bin histogram to be interpolated, and j is the index to the 8 bins of the resultant 8-bin gradient histogram. $Bin36_i$ represents the 36-bin gradient histogram excluding $Bin36_0$, $Bin36_9$, $Bin36_{18}$, and $Bin36_{27}$. F_{i_j} is the corresponding weighting factors that decides the weight of a bin in the 36-bin histogram to its two neighbours in the 8-bin histogram, and $Bin36_{i\_j}$ is the interpolated magnitude to be accumulated to the 8-bin gradient histogram.



(a)



(b)

Figure C-10: An example of the linear interpolation for 36-bin histogram.

Given an example bin with orientation of 100 ° (i=10) as shown in Figure C-10(a), 7/9 of its magnitude is accumulated to $Bin8_2$ with orientation of 90 ° (j=2) in the 8-bin histogram and 2/9 of its magnitude is accumulated to $Bin8_3$ with orientation of 135 ° (j=3), as shown in Figure C-10(b).

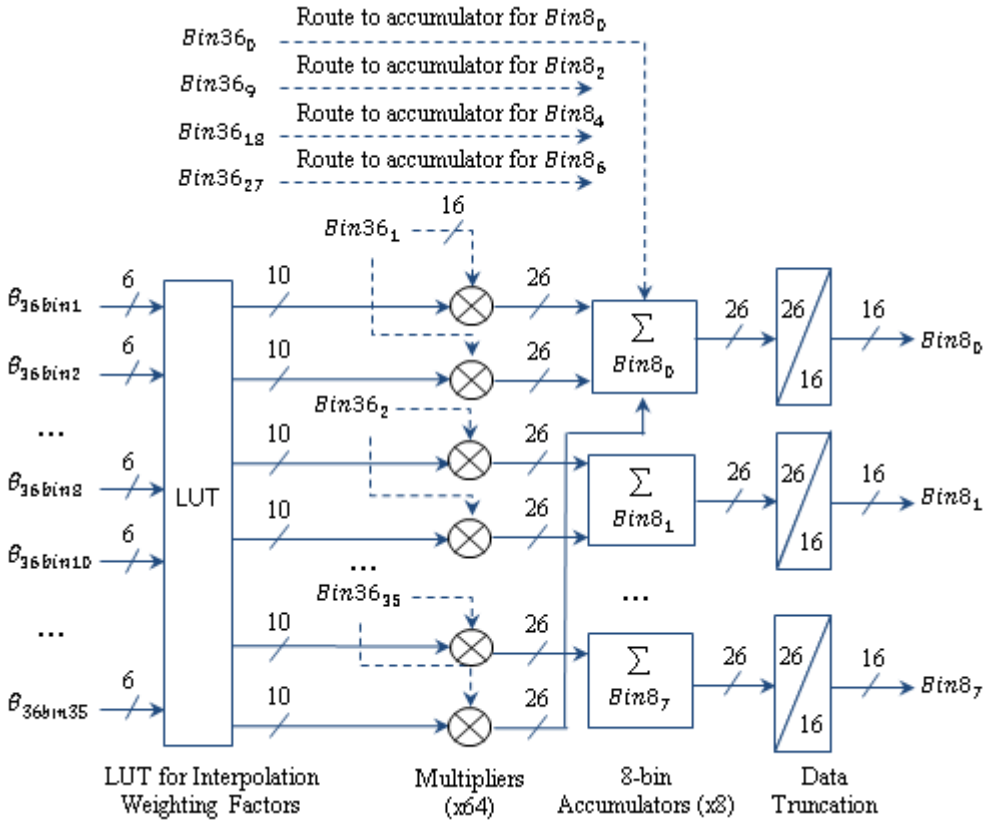


Figure C-11: Block diagram of Linear Interpolation module (pipeline stage 4).

The block diagram of this unit is shown in Figure C-11, which mainly consists of three parts: an LUT holding interpolation weighting factors, 64 multipliers, and 8 accumulators with each corresponds to a bin representing 45 °. In order to build up an efficient hardware architecture while preserving relatively high precision, F_{i_j} are scaled up by a factor of 1,024 with only the integer part preserved. It can be seen from Equation (C.2) that the calculation of F_{i_j} is only related to the distance from θ_{36bini} to θ_{8binj}, which represents the orientation of the 36-bin histogram and the resultant 8-bin histogram, respectively. Therefore, interpolation weighting factors F_{i_j} can be pre-calculated and saved in the LUT for fast indexing.

$$F_{i\_j} = \left[1 - \frac{\left|\theta_{36bini} - \theta_{8binj}\right|}{45}\right] \times 1024 \qquad \text{(C.2)}$$

Pre-calculating F_{i_j} turns interpolations into simple multiply-add operations. Besides, the 36 bins to be interpolated have no data dependency with each other and can be processed in parallel. It should be noticed that $Bin36_0$, $Bin36_9$, $Bin36_{18}$, and $Bin36_{27}$ are in the direction of $Bin8_0$, $Bin8_2$, $Bin8_4$, and $Bin8_6$, respectively, and are fully accumulated to the corresponding bins in the 8-bin histogram. Therefore, they are sent directly to the corresponding accumulators without interpolation, which save eight multipliers and hence there are 64 multipliers in total. Considering that F_{i_j} in Equation (C.1) are integers in range 0 to 1024, multiplications can be replaced by shifting, addition and subtraction operations, with which some precious hardware resources are saved, such as DSP48E1 on FPGA devices.

Descriptor Normalisation

This sub-module inputs the interpolated 8-bin histograms, and outputs the normalised descriptors. As shown in Figure C-12, it mainly consists of two identical Normalisation Units, a Threshold Bins unit and a multiplexer.

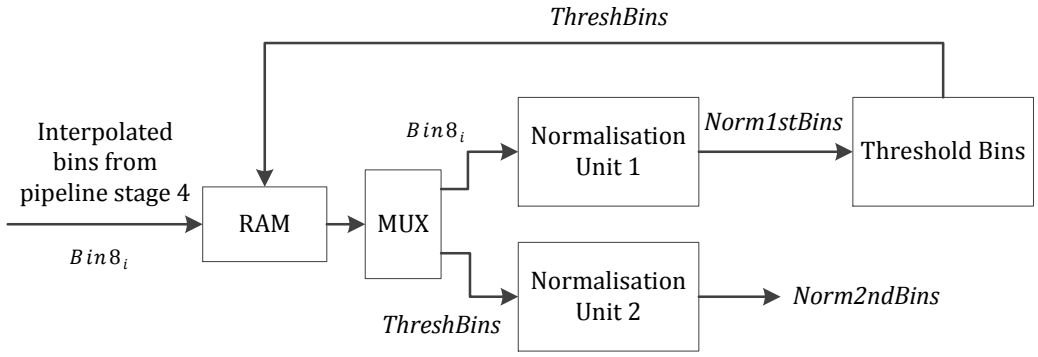


Figure C-12: Block diagram of the Descriptor Normalisation (pipeline stage 5).

As suggested in the SIFT algorithm, interpolated bins should be normalised twice and the second normalisation is performed to reduce the influence of large gradient magnitudes. Firstly, the interpolated bins ($Bin8_i$) are fed into Normalisation Unit 1 and are normalised to integers (*Norm1stBins*) in range 0 to 1023. Secondly,

Norm1stBins that is larger than the pre-defined threshold is forced to the value of the threshold in *Threshold Bins*. Finally, Normalisation Unit 2 performs the second normalisation and outputs *Norm2ndBins*, which are linked together to obtain the final descriptor of 72 dimensions.

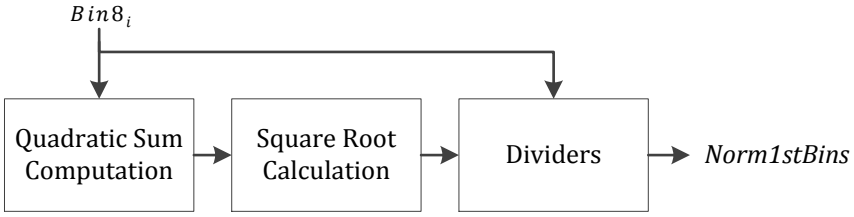


Figure C-13: Block diagram of the normalisation unit.

As shown in Figure C-13, each normalisation unit consists of three parts: a Quadratic Sum Calculator, a Square Root Calculator and Dividers. The SRT-based square root calculator and dividers are used to reduce the hardware resource usage.

Appendix D. The SIFT based Image Matching System

Camera Controller

The Camera Controller receives input images from the camera row by row, and further buffers the pixels in the input image buffer that is shared by the camera, the SIFT processing core and the USB. Because the image sensor and input image buffer works with two different clock domains of 40 MHz (PCLK) and 200 MHz, respectively, an asynchronous FIFO is employed in the exchange of data that transfers across different clock domains. Timing diagram of the OV9715 image sensor is given in Appendix F.

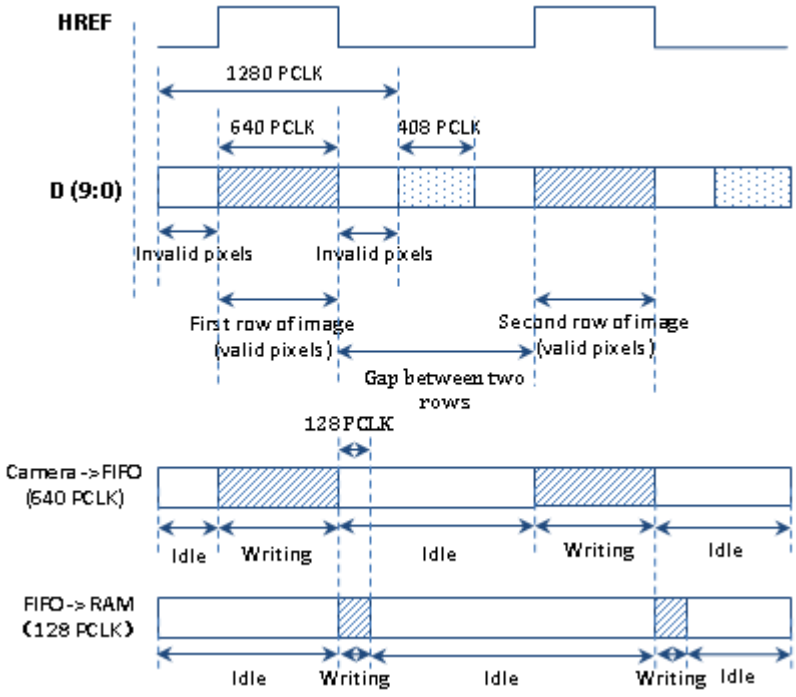


Figure D-1: Timing diagram showing that each line of pixels from the camera are first pushed into FIFO under PCLK domain, and then further buffered in RAM under clock of 200 MHz.

The 10-bit input pixels are synchronised by the rising edge of PCLK. Figure D-1 shows that it takes 640 PCLK clock cycles to push a row of valid pixels into one end of the FIFO for VGA sized image, and it requires 640 clock cycles of 200 MHz to drain the FIFO from the other end, which corresponds to 128 PCLK clock cycles. Because the gap between two rows of valid pixels is 1,048 PCLK clock cycles, the FIFO has been drained before the next row of pixels arrives.

USB Controller

Figure D-2 shows the block diagram of the USB Controller core that interfaces with the USB transceiver (CY7C68001) on the USB board. This core deals with the data transfer between the FPGA board and the host PC.

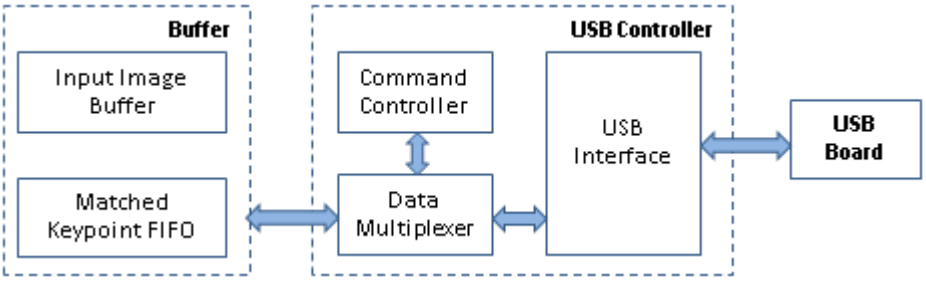


Figure D-2: Block diagram of the USB Controller with connection to buffers and the USB board.

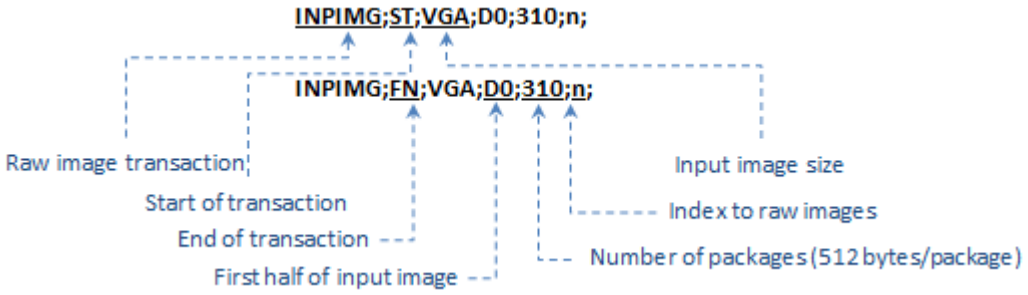
Two sets of data are transferred to PC via USB: the raw images, and the coordinates of matched keypoints. The raw images are sent to PC for the following two reasons.

- To visualise the matching results.
- The images can be processed by MALTAB model for comparison with the processing algorithm embedded in the FPGA device.

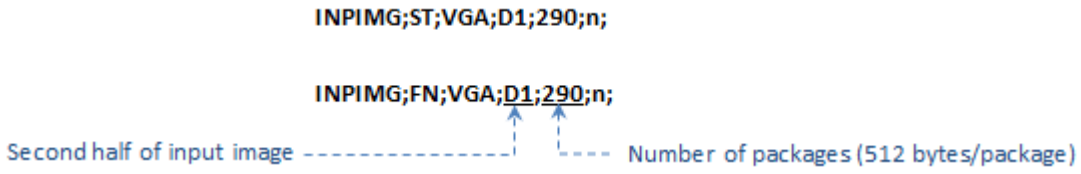
The USB Controller core mainly consists of the following three units:

- *Command Controller*: Generate two types of commands, which indicate that either the input image buffer has been filled with pixels, or coordinates of matched keypoints are ready to output. The commands are pushed into a FIFO to be accessed by *Data Multiplexer*.
- *Data Multiplexer*: Read control commands from *Command Controller* and select accordingly from *Input Image Buffer* and *Matched Keypoint FIFO*. The selected data is then routed to *USB Interface* for output. Header is attached to each set of data to avoid transaction errors.
- *USB Interface*: At the initialisation stage, this unit configures the USB board to high speed mode (480 Mbits/s) that sends 512 bytes in the transfer of each package. At the data output stage, this unit first sends an identification file to the PC, indicating the start of transaction for a set of data, followed by the data received from *Data Multiplexer* and ends up with the identification file that indicates the end of transfer.

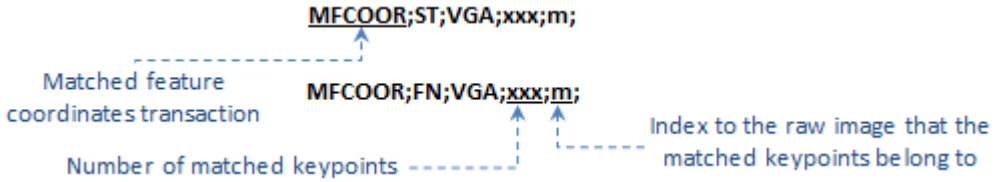
The format of identification files are given in Figure D-3. The PC can tell from the identification files the size of images, the index to the received data, and the number of packages to be received, especially for the coordinates of matched keypoints that vary with input images. Indexes n and m are important in that they tell which input image the received keypoints belong to.



(a) Identification files for the first half of the raw image.



(b) Identification files for the second half of the raw image.



(c) Identification files for the coordinates of the matched keypoints.

Figure D-3: Identification files attached to each set of data to be sent to PC.

As shown in Figure D-4, by concatenating the coordinates of a pair of matched keypoints as a single data, each pair of matches takes only 6 bytes and each package contains 85 pairs of matches plus two zero bytes. The PC reads from the identification file the number of matches and works out the number of packages to be received.

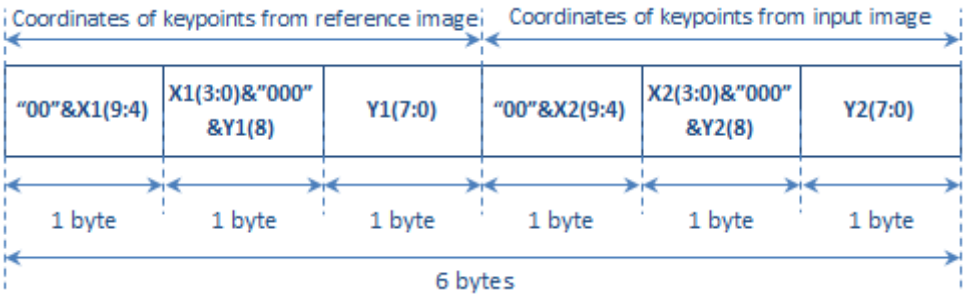


Figure D-4: Concatenation of matched keypoint coordinates.

Memory Controller

Due to the limited BRAM resources available on the FPGA device, the input image buffer is shared by the camera, the SIFT processing core and the USB. The Memory Controller core is designed to stop a frame from being over-written before it has been either processed by the SIFT processing core or sent to PC via USB. The input image buffer consists of two groups of RAMs with one for each half of the input frame. One group is being written by the camera while the other is being read by the SIFT processing core and the USB. Each group consists of two DPRAMs that together buffer half of an input frame. Port A of each DPRAM is shared between the camera front-end and the SIFT processing core and Port B is allocated to the USB only. The command for both DPRAMs within the same group is the same. Figure D-5 shows the command for one DPRAM in each group.

In this design, each group contains two DPRAMs, so pixels can be accessed from two channels (Port A) concurrently by the SIFT processing core. The input frames are continuously routed to and from the input image buffer under the control of the Memory Controller and only the latest frame can be stored on the buffer.

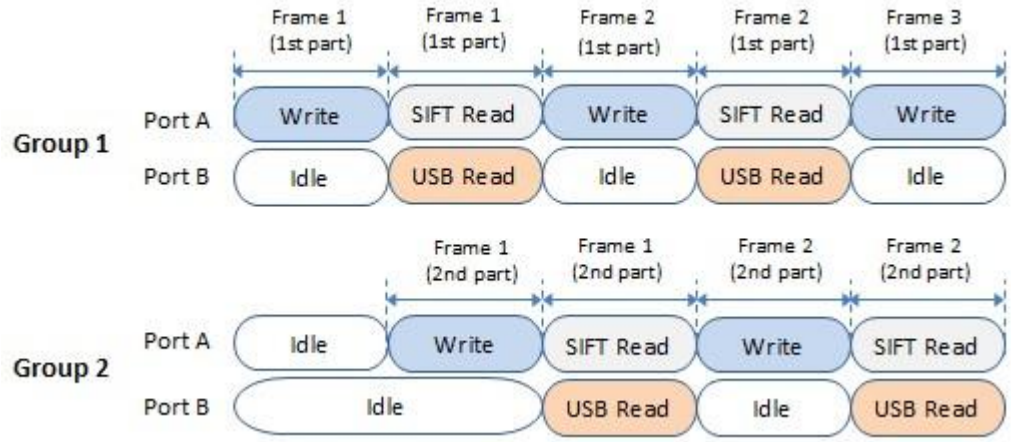
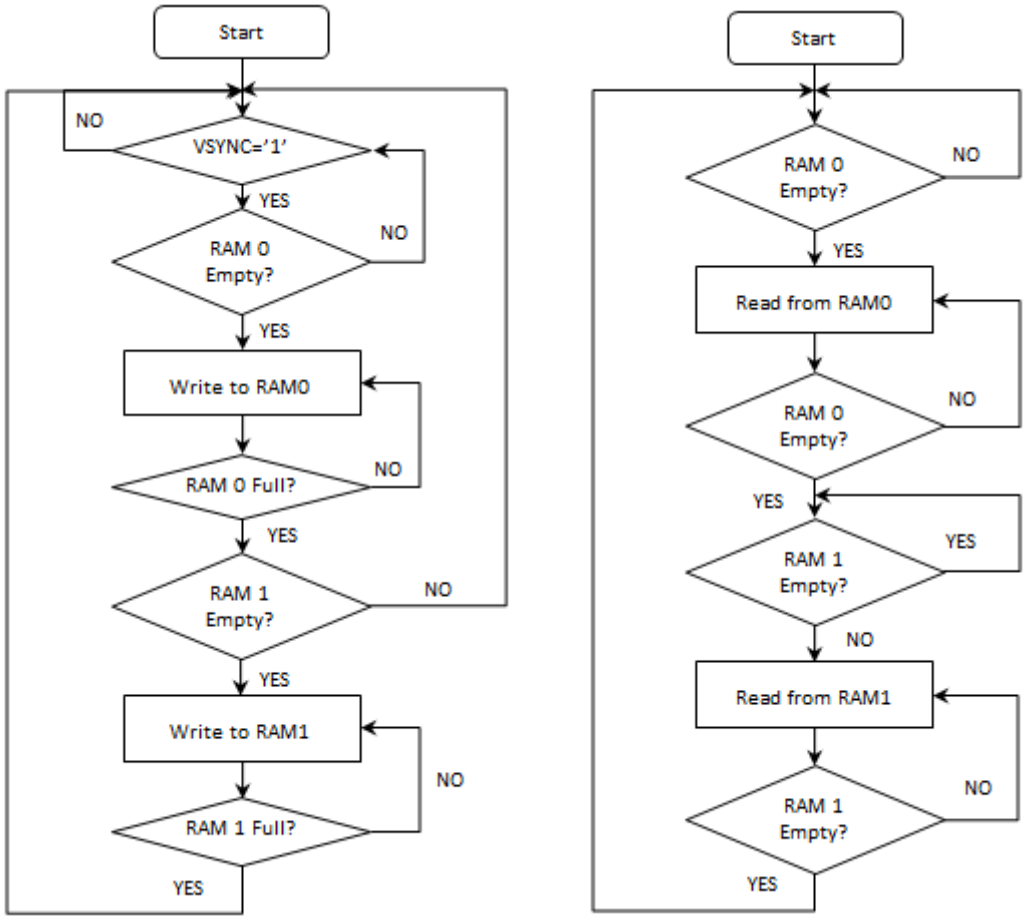


Figure D-5: Command for input image buffer access.

Figure D-6 shows the flowchart for writing and reading the input image buffer by the camera and the SIFT processing core, respectively. The status of RAM0 is checked when a new frame arrives (VSYNC='1'). If RAM0 is empty before the first row synchronisation signal (HREF) is asserted, it indicates that RAM0 is ready to accept the first part of a new frame. Otherwise, the coming frame is abandoned as the data from the previous frame is still waiting to be sent to PC via the USB. The status of RAM1 is checked when RAM0 has been filled up with the first part of an input image. If RAM1 is empty before the next row synchronisation signal (HREF) is asserted, it indicates that RAM1 is ready to accept the other part of the new frame and SIFT processing core is enabled to process the first half frame.

It should be noticed that the SIFT processing core is not enabled immediately after RAM0 has been filled with data due to the possibility that the second half of frame may be abandoned. Once RAM1 has been filled up with data, the SIFT processing core is enabled to process the second half frame. Status checking for RAM0 and RAM1 is necessary in that the previous frame may be overwritten by the new one before it has been fully accessed by the USB.



(a) (b)

Figure D-6: (a) Flowchart for writing input image buffer by the camera. (b) Flowchart for reading input image buffer by the SIFT processing core.

Display Controller

A MATLAB based software model is written to run on a PC to visualise the results of the developed image matching system. The flowchart of the MATLAB based display controller is given in Figure D-7.

The software model communicates with the FPGA board through the USB link, and performs the following tasks:

- Keep retrieving data package by package from FPGA until a valid identification file is received, which indicates that a set of data is ready to be transferred from FPGA to PC.
- Buffer the raw images if the entire frame has been correctly received.
- Buffer the coordinates of matched keypoints and display the matching results.

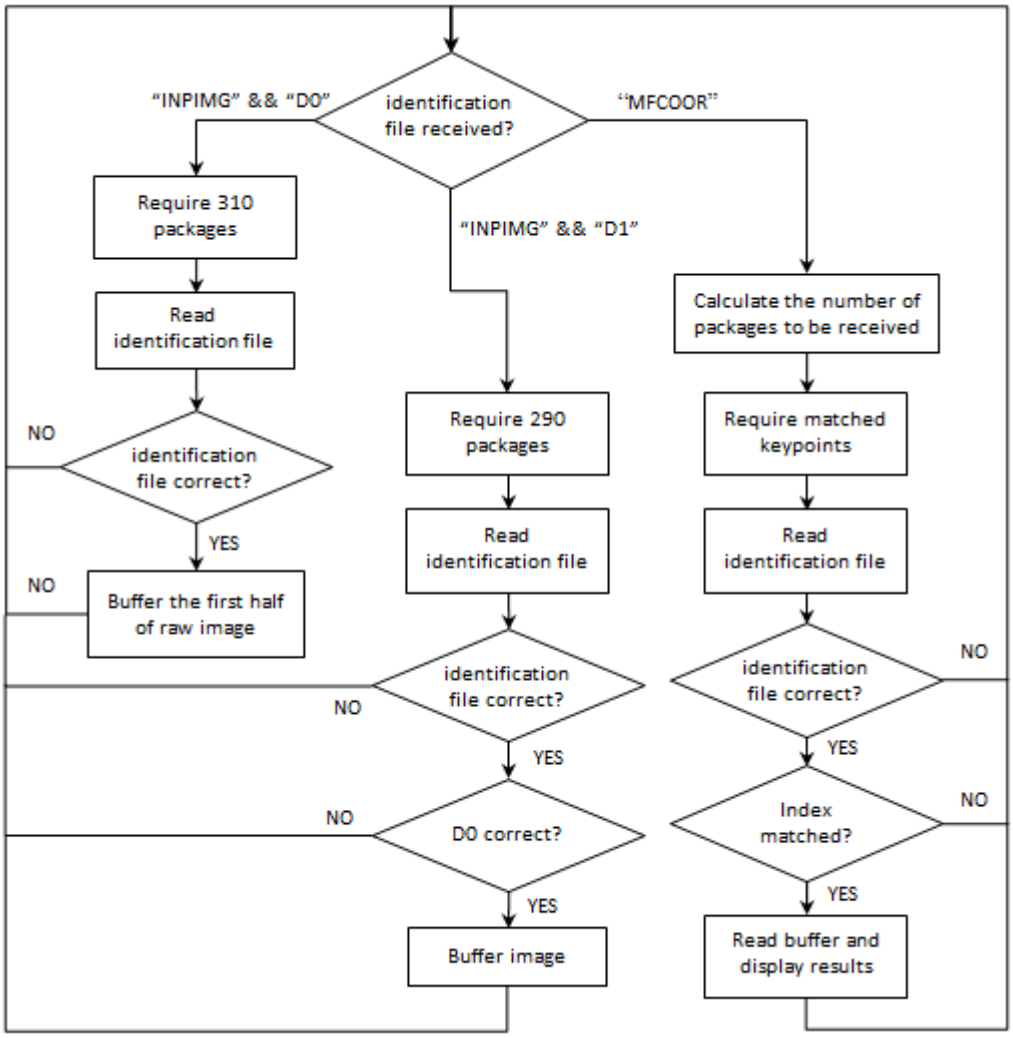


Figure D-7: Flowchart for the Display Controller.

Appendix E. Avnet FMC Module

As shown in Figure E-1, the Dual Image Sensor FMC Module is a low pin count (LPC) FMC module containing interfaces intended for video processing. This module contains no processing intelligence and requires that it be plugged into a compatible baseboard for power, control and data processing.

Figure E-1: The left image shows the top of the Avnet FMC module. The right image is for the bottom.

All the camera functions can be configured through I^2C interface by writing in some registers, such as the frame rate and size of input images. The registers are accessed by the I^2C bus, but the actual protocol used for communication is the Serial Camera Control Bus (SCCB) interface which is for some mode the same as I^2C protocol. In the I^2C protocol, two pins are used. One is the clock (SCL) and the other is the data (SDA). The SCCB protocol consists of two signals, which is the single-directional control signal (SIO_C) and bi-directional data signal (SIO_D), respectively. The SIO_C must be driven by the master device, while SIO_D can be driven by both master and slave device. As shown in Figure E-2, FPGA and FMC module with image sensor acts as the master and slave device, respectively.

Figure E-2: Function block diagram for the 2-wire SCCB.

Figure E-3 shows the detailed block diagram of the slave device. We configure image size and video frame rate by writing in registers of image sensor that is connected to port 2 of the I^2C multiplexer. The clock period of the input clock (XCLK) of the camera is set by writing to registers of video clock generator that is connected to port 3 of the I^2C multiplexer. Configuration of other devices remains as default. The corresponding I^2C addresses are given in Table E-1.

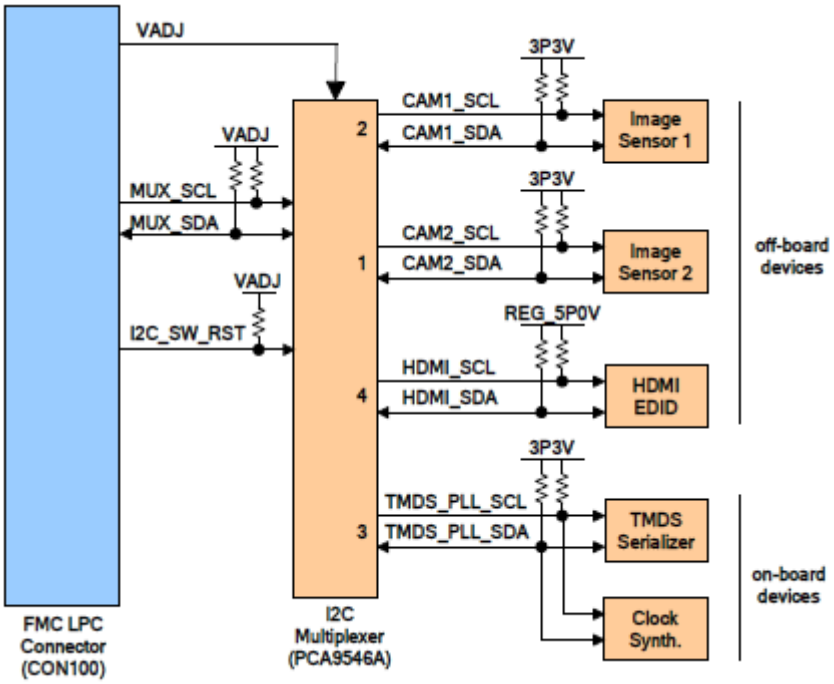


Figure E-3: Block diagram of slave device for I^2C peripheral configuration.

Table E-1: I^2C address.

Device	I^2C Address
I^2C Multiplexer	0xE0
Image Sensor	0x60
Video Clock Synth.	0xC8

As shown in Figure E-4, the basic element of the data transmission using the SCCB protocol is called a phase, and each write transmission cycle consists of three phases. Each phase consists of 9 bits, where the ninth bit is a Don't-Care bit or an NA bit, depending on whether the data transmission is a write ('0') or read ('1'). The IP address corresponds to the I^2C address of devices, and the sub-address is the address of the register to be written to. The SCCB protocol is implemented using VHDL and the timing diagram is shown in Figure E-5. Detailed configuration parameters are given in Table E-2.

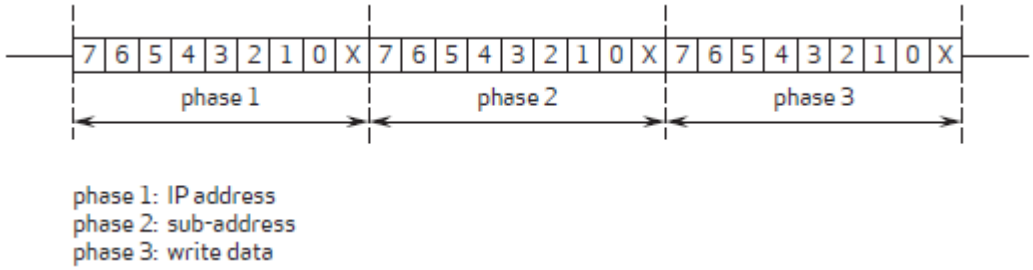


Figure E-4: The 3-phase write transmission cycle.

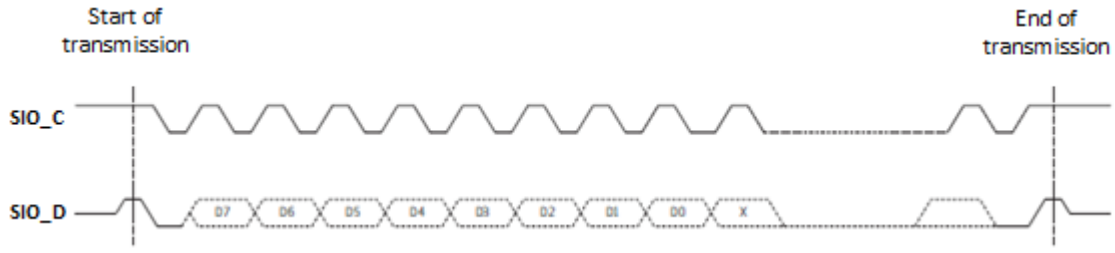


Figure E-5: Timing diagram of SCCB configuration.

Table E-2: Configuration of FMC Module with Image Sensor

Device	Function	Register	Value
Video Clock Generator (0x60)	Set the clock period of the input clock (XCLK) of the camera to 12MHz.	0x24	0x6D
		0x26	0x12
		0x27	0x12
		0x28	0xFF
		0x29	0x80
		0x2A	0x02
		0x2B	0x07
Image Sensor (0x60)	Set the size of the input image to 640x480.	0x17	0x25
		0x18	0xA2
		0x19	0x01
		0x1A	0xCA
		0x03	0x0A
		0x32	0x07
		0x98	0x40
		0x99	0xA0
		0x9A	0x01
		0x57	0x00
		0x58	0x78
		0x59	0x50
		0x4C	0x13
		0x4B	0x36
		0x3D	0x3C
		0x3E	0x03
		0xBD	0x50
		0xBE	0x78
	Set the video frame rate to 30fps, assuming the clock period of the input clock (XCLK) of the camera has been set to 12MHz.	0x5C	0x19
		0x5D	0x00
		0x11	0x00

Appendix F. OmniVision OV9715 Image Sensor

The OV9715 image sensor is one megapixel CMOS image sensor that has an image array capable of providing full-frame, sub-sampled or windowed 8-bit/10-bit images in raw RGB format. The sensor delivers XWGA (1280x800) resolution video at 30 fps and the maximum image transfer rate for 640x400 resolution video is 60 fps. In our system, the image sensor is configured to deliver 640x480 resolution video at 30 fps by truncation, as shown in Figure F-1.

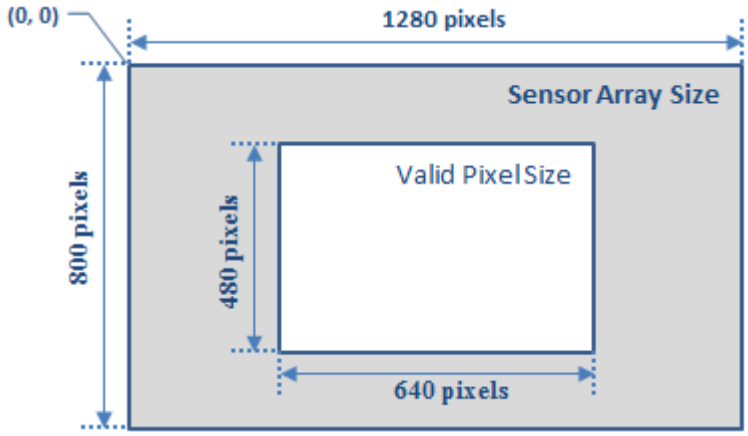


Figure F-1: Sensor array size (1280x800) and valid pixel size (640x480).

Detailed timing diagram is shown in Figure F-2, where VSYNC indicates the starting of a new frame and HREF indicates when the pixels are valid. The image sensor array is accessed row by row, and HREF acts as the row synchronisation signal. PCLK is the clock signal that is configured to 40 MHz, and all the other signals are synchronised by the rising edge of PCLK. D is the 10-bit input data.

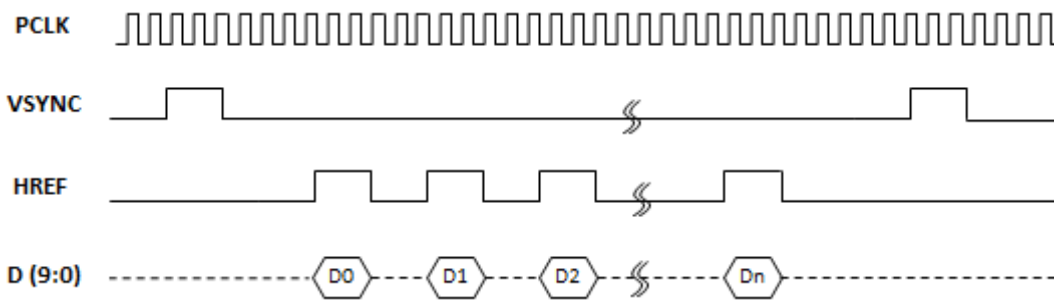


Figure F-2: Timing diagram of the image sensor.